Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Számítástudományi és Információelméleti Tanszék

Székelyföld énekes és hangszeres zenekultúrájában rejlő ritmikai összefüggések vizsgálata adatbányászati módszerekkel

Diplomaterv

Készítette
Nyitrai József

Konzulens
Juhász Zoltán

2013. május 26.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Nyitrai József*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2013. május 26.

_____

Nyitrai József
hallgató

Kivonat

Diplomamunkám írása során a célom az volt, hogy az adatbányászat kapcsolódó módszereit bemutatva egy népzenei kottákból álló adatbázis felett működő, rejtett összefüggések keresésére alkalmas szoftvert specifikáljak és hozzak létre.

Az általam vizsgált attribútum a zene ritmusa, melynek összehasonlíthatóvá tételéhez különböző ritmikai metrikákat kellett definiálni. Ezek segítségével az előforduló ritmusok (pontosabban az őket reprezentáló vektorok) között ritmikai távolságok határozhatóak meg, melyek alapjául szolgálnak a rejtett összefüggések feltárására szolgáló klaszterező (valamilyen szempontból hasonló elemeket tartalmazó, de előre nem definiált osztályokat kereső) algoritmusoknak.

Az adatbányászati feladatok, így a klaszterezés is erősen függ a vizsgálandó adatbázis speciális tulajdonságaitól, nincs mindig jól működő, általános módszer. Emiatt meg kell vizsgálni az alkalmazási terület specifikus jellemzőit, és több olyan módszert is javasolni, mely megoldást nyújt a feladatra.

Az osztályokon belüli elemek közös tulajdonságának vizsgálata adja meg a választ arra, hogy milyen összefüggés alapján kerültek egy csoportba az adott elemek. A klaszterek elemeik egymástól való távolságait tükröző vizualizációs eszközök segítségével tehetőek szemléletessé.

Az átfogóbb megértés érdekében a kapott eredményeket össze kell vetni a dallamvonalat, mint fő attribútumot vizsgáló korábbi kutatások eredményeivel. A feladatot magába foglaló projekt hosszú távú célja ugyanis a világ népzenei kapcsolatainak átfogó feltárása, melynek elengedhetetlen feltétele a többszempontú megközelítés és a különböző módszerek eredményeinek együttes értelmezése.

Az elsődlegesen használt adatbázis az erdélyi Székelyföldről származó hangszeres és énekelt népzenei gyűjteseket tartalmazza. Ez néhány ezer speciálisan kódolt kottát jelent, melyen az algoritmusok működése tesztelhető és értékelhető, majd a későbbiek során könnyedén bővíthető.

Abstract

While working on this thesis, my aim was to demonstrate the related methods of data mining and to specify and develop a software that is able to find hidden connections in a database that contains folksong sheets.

The chosen attribute is the rhythm of the songs, which needs us to have well-defined rhythmic metrics. With these (or more precisely with the vectors representing these rhythms) we can determine rhythmic distances between songs, which are the underlying metrics for the hidden connection seeker clustering (partitioning objects into groups in such way that elements belonging to the same group are more similar to each other than to those in other groups) algorithms.

The data mining tasks, including clustering analysis strongly depend on the data they examine. There is no method which works well generally for each dataset so we have to study the specific properties of the field we would like to study (music theory and systematization), and also this is the reason we present more algorithms that can solve the problem.

Investigating the common properties of the elements that belong to same group can reveal the reason why these objects got to the same cluster. The clusters can be comprehensible using graphical methods that are based on the distance between the elements of the given cluster.

To be sure that we properly interpret the results, we have to compare them with the results of the former analysis based on melodic chains. The long-term aim of the project that I would like to add more with my thesis is to discover hidden relationships in the folk music heritage of the world. To hit this aim it is necessary to study the database in every aspect.

The database used in this thesis contains data from vocal and instrumental folksong collections of the Transylvanian region, Szeklerland. This means a few thousands of special encoded sheet music, which is enough to test and examine the algorithms and can also be easily extended in the future.

Bevezető

Bartók Béla írja 1937-ben: *„...az a gyanúm, hogy a földkerekség minden népzenéje, ha elegendő népzenei anyag és tanulmány áll majd rendelkezésünkre, alapjában véve visszavezethető lesz majd néhány ősformára, őstípusra, ős stílus-fajra."* [1]

A történelmi Magyarországon a 19. század végén megkezdődő és azóta is tartó tudományos célú és formájú népzenei gyűjtések óriási mennyiségű népzenei adatot szolgáltattak, aminek elemzésére a hagyományos megközelítés már nem bizonyul elegendőnek. A nagyméretű adathalmaz feldolgozásához az időközben hatalmasat fejlődő informatika tudományához, azon belül is az adatbányászathoz fordulhatunk segítségért. Azért is érdemes lehet ezt tennünk, mert a gépi módszerek olyan dolgokra is ráirányíthatják a figyelmet, melyek eddig rejtve maradtak.

Juhász Zoltán, az MTA Műszaki Fizikai és Anyagtudományi Kutatóintézetének kutatómérnökeként évek óta foglalkozik a népzene rejtett összefüggéseinek számítógépes vizsgálatával, modellezésével, elemzésével. Mindemellett a magyar pásztorság hangszeres és énekes zenei hagyományának szenvedélyes kutatója, előadója, tanítója. Nevéhez fűződik a jelen téma kiindulásaként tekinthető dallamvonal-alapú osztályozás elkészítése.

A több tudományos diszciplínát érintő projekt hosszú távú célkitűzése a Kárpát-medence népzenéjének, azon belül is főképp a magyar népzenének az elhelyezése a világ népzenei térképén, illetve a bartóki cél minél közelebbi elérése. Ehhez a különböző zenei attribútumok alapján készülő csoportosításokat össze kell hasonlítani és levonni a közös következtetéseket.

A dolgozatom első szakaszában szó esik az adatbányászat jelenlegi helyzetéről, lehetőségeiről, a kapcsolódó algoritmusokról és azok alkalmazhatóságáról a konkrét feladatra. Ezután az alkalmazási terület múltjának és jelenének rövid összefoglalása következik, majd a speciális, ritmus mérésére alkalmas módszereket ismertetem azok egymástól való eltéréseinek hangsúlyozásával.

Az ezt követő szakaszban szó esik az implementációról, kitérve a fejlesztési folyamatban előbukkanó problémák megoldásaira. A dolgozatot az eredmények bemutatásával és a továbbfejlesztési lehetőségek vizsgálatával zárom.

1. fejezet

Adatbányászati háttér

Első feladatunk, hogy áttekintsük az adatbányászat jelenlegi helyzetét, megtaláljuk a projekt szempontjából releváns lehetőségeit, majd ismertessük a problémára megoldást kínáló algoritmusokat.

1.1. Az alkalmazott adatbányászat

Az adatbányászatot sokféleképp lehet definiálni. A legelterjedtebb megfogalmazás szerint olyan újszerű, érvényes és korábban ismeretlen tudás kinyerése nagyméretű adathalmazból, mely nem triviális, hasznos és valamilyen módon magyarázható is [2]. Mint formula, nem a legszerencsésebb kifejezés, hiszen míg a szénbányászaton a szén kitermelését értjük, az adatbányászat nem új adatot, hanem a nyers adatból tudást hoz létre.

Az alkalmazott adatbányászat egy multidiszciplináris terület, hiszen az alapjait adó tágabb értelemben vett matematikai ágak (statisztika, valószínűség-számítás, lineáris algebra, algoritmus-elmélet, mesterséges intelligencia, stb...) mellett nem hagyható figyelmen kívül az adathalmaz által reprezentált entitásokhoz kapcsolódó meglévő tudásunk sem, hiszen biológusok, csillagászok, bankárok nap mint nap bár, de különbözőképp használják és magyarázzák eredményeit.

A dolgozatom egy etnomuzikológus által felvetett kérdésekre keres adatbányászaton alapuló, de elsősorban népzenekutató szakember által értelmezhető válaszokat, tehát a cél, hogy a népzenekutatás számára állítsunk elő új tudást.

1.2. A tudásfeltárás folyamata

A tudáskinyerési folyamat jól definiált, egymást követő lépések sorozata. A kitűzött célok elérését segíti, ha követjük ezeket a lépéseket, melyek a korábbi kutatások tapasztalatain alapulnak.

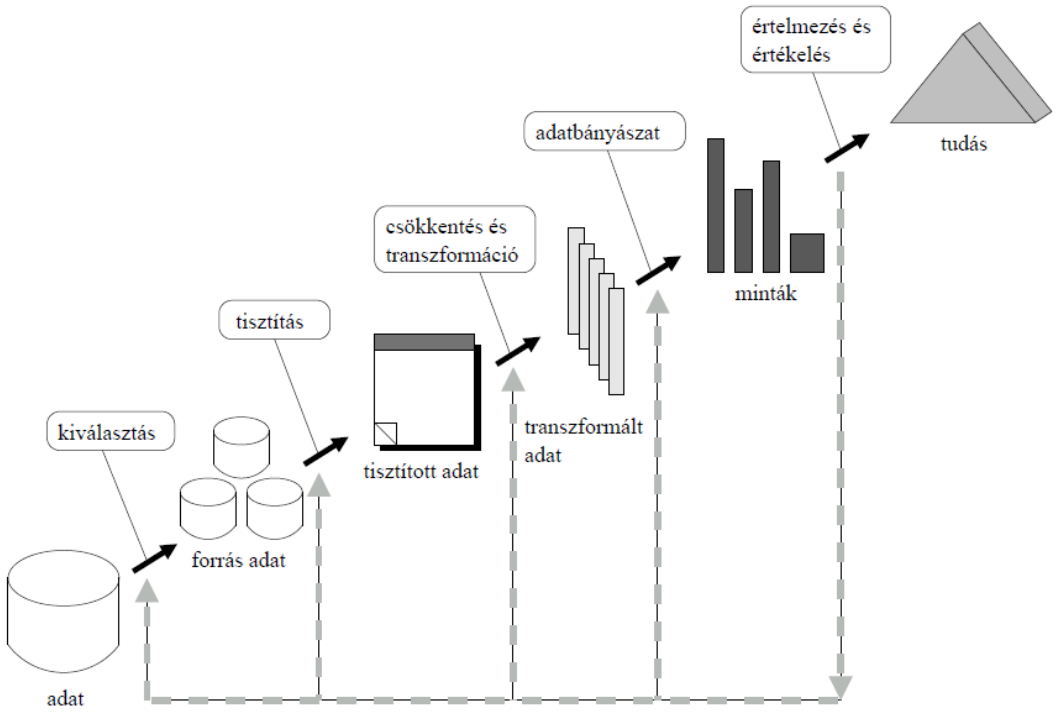
A témával foglalkozó szakirodalom [2, 3] különböző felosztásokban (némely lépéseket egybevonva vagy több pontra bontva), de a következő fázisokat sorolja fel (1.1 ábra):

1. **Az alkalmazási terület feltárása és megértése**: a fontosabb előzetes ismeretek begyűjtése, a felhasználási célok meghatározása
2. **A céladatbázis létrehozása**: a használni kívánt adatbázis kiválasztása, amelyből a tudást ki akarjuk nyerni
3. **Az adatok előfeldolgozása**: téves bejegyzések eltávolítása, hiányos mezők kitöltése, zajszűrés (adattisztítás)
4. **Adattér csökkentés**: az adatbázisból a cél szempontjából fontos attribútumok kiemelése
5. **Az adatbányászati algoritmus típusának kiválasztása**: a feladatra megoldást nyújtó módszerek áttekintése és a használni kívánt típus meghatározása
6. **A megfelelő algoritmus kiválasztása**: konkrét algoritmus megadása idő-, és tárigény, előnyök és hátrányok alapján
7. **Az algoritmus futtatása**: az előkészített adatok elemzése a kiválasztott algoritmussal
8. **A kinyert tudás értelmezése**: a kinyert összefüggés vizsgálata az alkalmazási terület kontextusában, esetleges visszalépés finomítás céljából
9. **A megszerzett tudás megerősítése**: az eredmények összevetése az elvárásokkal, előzetes ismeretekkel

A jelen adatbányászati feladat számára adott volt a kották alkotta adatbázis, melyben sem zajszűrésre, sem téves vagy hiányos mezők javítására nem volt szükség. A dallamra vonatkozó információk elhagyásával megkaptuk a használni kívánt, csak ritmikai tulajdonságokra szűkített adatbázist. Az osztályozás előre nem definiált csoportok létrehozásával történik, azaz klaszterezésre van szükségünk. A rendelkezésre álló algoritmusokat a következő pontokban vizsgálom meg.

1.3. A klaszterezés

A klaszterezés egy olyan dimenziócsökkentő eljárás, melynek során az elemeket homogén csoportokba, úgy nevezett klaszterekbe soroljuk [4]. Az egyes klasztereken belüli elemek valamilyen szempontból hasonlítanak egymáshoz és különböznek a



1.1. ábra. *A tudásfeltárás folyamata*

többi klaszter elemeitől. A csoportosítás alapját különböző távolságmértékek képezik. Formálisan, a klaszterezési struktúra az elemeket tartalmazó S univerzum olyan részhalmazainak halmaza ($C = C_1, .., C_k$), hogy $S = \bigcup_{i=1}^{k} C_i$ és $C_i \cap C_j = \emptyset, \forall i \neq j$-re.

Ha S elemeinek száma n, és k darab csoportra szeretnénk bontani, akkor a lehetséges csoportosítások számát a Stirling számok adják meg:

$$S_n^{(k)} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^n \tag{1.1}$$

mely, $n = 25$ és $k = 5$ esetén már 10^{15} nagyságrendű, ráadásul ha még a k-t sem tudjuk, akkor még ilyen kisméretű adathalmaz esetén is 10^{18}-nál is több lehetőségünk van erre. Általában jóval nagyobb n-nel és k-val kell dolgoznunk, ami mutatja a probléma során előálló keresési tér hatalmasságát.

A klaszterezés egy nagyobb adathalmaz struktúrájának feltárására is használható, hogy egy áttekinthető képet kapjunk arról, milyen objektumok találhatóak benne. Ehhez az azonos csoportba került elemek közös tulajdonságait kell meghatároznunk.

A klaszter-analízis két fő ága a hierarchikus és a nem hierarchikus klaszterezés. Hierarchikus esetben az új klasztereket a korábbi klaszterek alapján határozzuk meg, ezzel szemben a nem hierarchikus módszerek egyszerre határozzák meg az összes

klasztert.

1.3.1. A klaszterek és a klaszterezés jellemzői

Egy klaszterezési algoritmus futása közben, illetve az eredmények összehasonlítása során alapvető kérdés, hogy hogyan értékelünk egy adott klaszterezést. Hogy ezt megtehessük, definiálnunk kell néhány klaszter-, illetve klaszterezésjellemzőt.

A C klaszter elemeinek számát $|C|$ jelöli. A klaszter nagyságának jellemzésére szolgál az átmérő fogalma ($D(C)$). Ezt többféleképp is definiálhatjuk:

- A klaszter elemei közötti maximális távolság alapján:

$$D_{max}(C) = \max_{\substack{p \in C \\ q \in C}} d(p, q) \tag{1.2}$$

- A klaszter elemei közötti átlagos távolság alapján:

$$D_{avg}(C) = \frac{\sum_{\substack{p \in C \\ q \in C}} d(p, q)}{|C|^2} \tag{1.3}$$

- Ugyanez, ha nem vesszük számításba az elemek önmaguktól való 0 távolságát:

$$D'_{avg}(C) = \frac{\sum_{\substack{p \in C \\ q \in C}} d(p, q)}{\binom{|C|}{2}} \tag{1.4}$$

Ha a klaszterezendő elemek vektortérben értelmezhetőek, akkor beszélhetünk a klaszter középpontjáról ($\vec{m}_C$), illetve sugaráról (R_C), mint klaszterjellemzőkről. Ezeket így definiáljuk:

$$\vec{m}_C = \frac{\sum_{p \in C} \vec{p}}{|C|} \tag{1.5}$$

$$R_C = \frac{\sum_{p \in C} |\vec{p} - \vec{m}_C|}{|C|} \tag{1.6}$$

Gyakran előfordul, hogy egy algoritmus következő állapotának meghatározásához meg kell találnunk egy adott célfüggvényt minimalizáló klaszterpárt. Ilyen célfüggvény lehet a klaszterek közötti aktuális távolság, melyet többféleképpen is meghatározhatunk:

- A két klaszter elemeinek páronként számított távolságainak minimuma:

$$d_{min}(C_i, C_j) = \min_{\substack{p \in C_i \\ q \in C_j}} d(p, q) \tag{1.7}$$

- A két klaszter elemeinek páronként számított távolságainak maximuma:

$$d_{max}(C_i, C_j) = \max_{\substack{p \in C_i \\ q \in C_j}} d(p, q) \tag{1.8}$$

- A két klaszter elemeinek páronként számított távolságainak átlaga:

$$d_{avg}(C_i, C_j) = \frac{\sum_{\substack{p \in C_i \\ q \in C_j}} d(p, q)}{|C_i| * |C_j|} \tag{1.9}$$

- A két klaszterközéppont távolsága:

$$d_{mean}(C_i, C_j) = |\vec{m}_i - \vec{m}_j| \tag{1.10}$$

- Az egyesítésükkel keletkező klaszter átmérője:

$$d_D(C_i, C_j) = D(C_i \cup C_j) \tag{1.11}$$

Amennyiben már van egy klaszterezésünk, és azt össze szeretnénk hasonlítani egy másikkal, akkor egy adott klaszterezési struktúrára is definiálnunk kell mérőszámokat. A következő függvények minimalizálása a legelterjedtebb klaszterező algoritmusok esetén:

- Hibaösszeg (klasztereken belüli elemek klaszterközéppontól való távolságainak összege):

$$\sum_{i=1}^{k} \sum_{p \in C_i} |\vec{p} - \vec{m}_{C_i}| \tag{1.12}$$

- Négyzetes hibaösszeg (ugyanaz, mint az előző, csak négyzetesen - a nagy távolságok súlyának növelése céljából):

$$\sum_{i=1}^{k} \sum_{p \in C_i} (|\vec{p} - \vec{m}_{C_i}|)^2 \tag{1.13}$$

- Klasztereken belüli távolságösszegek összege:

$$\sum_{i=1}^{k} \sum_{\substack{p \in C_i \\ q \in C_i}} d(p,q) \tag{1.14}$$

- Maximális átmérő:

$$\max_{i \in \{1,\ldots,k\}} D(C_i) \tag{1.15}$$

- Átlagos átmérő:

$$\frac{\sum_{i \in \{1,\ldots,k\}} D(C_i)}{k} \tag{1.16}$$

Az utolsó két mértékben használt $D(C)$ klaszterátmérő természetesen mindhárom fentebb definiált módon ($D_{max}(C)$: 1.2-es képlet, $D_{avg}(C)$: 1.3-os képlet, illetve $D'_{avg}(C)$: 1.4-es képlet) behelyettesíthető.

A következő fejezetek algoritmusainak ismertetéséhez feltétlenül szükséges volt a fenti metrikák pontos definiálása, hiszen az eljárások szinte minden lépésük során használják valamelyiküket.

1.3.2. Hierarchikus klaszterezési módszerek

Ezek az eljárások közösek abban, hogy iteratívan, fentről lefelé vagy lentről felfelé építkezve a klaszterezés aktuális állapota alapján határozzák meg annak következő felosztását [5]. Az építkezés stratégiája alapján a következőképpen lehet őket alosztályokra bontani:

- Agglomeráló (összegyűjtő) hierarchikus klaszterezés: Kezdetben minden objektum egy önálló klasztert alkot, majd ezek összefésülésével jutunk el a kívánt célhoz.

- Divizionáló (felosztó) hierarchikus klaszterezés: Kezdetben minden objektum egy nagy közös klaszterba tartozik, majd ennek rekurzív felosztásával kapjuk meg a végső állapotot.

Az összefésülés, illetve a felosztás valamilyen távolságmérték alapján történik. Minden lépésben egy előre megadott jósági kritériumot optimalizálunk. A távolságértékek alapján történő következő lépés meghatározása alapján további felosztásokat tehetünk a hierarchikus klaszterező eljárásokra:

Single-linkage klaszterezés

Ebben az esetben két klaszter távolságát a bennük lévő elemek páronkénti távolságainak minimumaként definiáljuk (1.7-es képlet).

Gráfelméleti szempontból nézve (egy teljes gráfban a pontok a klaszterezendő elemek, az élsúlyok pedig a pontokhoz tartozó elemek közötti távolságok) ez a módszer egy minimális feszítőfát fog találni, ha a klaszterek számát 1-re állítjuk. Ha k darab csoportot szeretnénk kapni, akkor ezt a minimális feszítőfa $k-1$ darab legnagyobb súlyú élének elhagyásával kaphatjuk meg. Az így keletkezett komponensekben található elemek kerülnek egy klaszterbe.

A módszernek komoly hátránya az úgy nevezett lánceffektus: néhány hídként viselkedő egymáshoz közeli pont összekapcsolhat egyébként jól elkülönülő klasztereket. További megfontolandókat vet fel az elkülönülő pontok (úgynevezett outlierek) jelenléte az adatbázisban.

Complete-linkage klaszterezés

Ez a megközelítés is a klasztereken belüli elemek páronkénti távolságából indul ki, de minimum helyett azok maximumát használja a klasztertávolságok meghatározására (1.8-as képlet).

Gráfelméleti szempontból itt most maximális, adott feltételnek megfelelő teljes részgráfok, azaz részgráfként előálló klikkek keresése a cél, melyek meghatároznak egy klasztert. A feltétel az, hogy minden szomszéd egy számított maximális távolságon belül legyen.

A kapott klaszterek kompaktsága miatt gyakran ez adja a legjobb eredményeket, hiszen kiküszöböli a single linkage klaszterezésnél előforduló lánceffektust. Jellemző rá, hogy a klaszterek átmérői megközelítőleg hasonló méretűek.

Average-linkage klaszterezés

Ez a módszer két klaszter távolságát elemeik páronkénti távolságainak átlagaként definiálja (1.9-es képlet). Hátránya, hogy „hosszúkás” klasztereket szétbonthat, illetve szomszédos „hosszúkás” klasztereket összevonhat.

Ward módszere

Ennél a módszernél is kezdetben minden elem külön klaszterbe tartozik, a négyzetes hiba így 0. Ezután minden lépésnél a négyzetes hibát próbálja minimalizálni azzal, hogy azt a két klasztert egyesíti, amelyek összevonása a legkisebb négyzetes hibanövekedést okozza. Sajnos itt sem garantált, hogy globális helyett nem lokális minimumot találunk.

Minden további felosztás egy nagyobb távolságot eredményez a klaszterek között, mint az előző felosztás maximális távolsága. A leállási feltétel így lehet egy maximális új távolság is a k számú klaszter elérése mellett.

A hierarchikus módszerek közös hátránya a rossz skálázhatóság és a nemlineáris lépésszám (legjobb esetben is ($O(n^2)$, de akár $O(2^n)$ is lehet, ahol n az elemek száma), melynek következménye nagyméretű adathalmaz esetén a túl sok fájlművelet. Súlyos hátrány még a visszalépési képesség hiánya.

1.3.3. Particionáló módszerek

Ezek a módszerek abban közösek, hogy egy kezdeti k diszjunkt particionálás után az elemeket ide-oda mozgathatják a klaszterek között, minden lépés során ügyelve arra, hogy egy előre megadott metrikát (ilyen lehet az 1.12-1.16-os függvények bármelyike) minimalizálják vagy maximalizálják. Amennyiben több lehetőség is kínálkozik a célfüggvény javítására, a legjobbat választják ezek közül. Akkor állnak le, ha már nem lehet több olyan lépést tenni, ami javítaná az aktuális klaszterezés jóságát.

k-means klaszterezés

A k-means az egyik legrégebbi és legegyszerűbb klaszterező algoritmus, mely vektortérben elhelyezkedő elemek esetén használható. Menete a következő: kezdetben választunk k darab véletlen elemet, melyek reprezentálják a k darab klasztert. Ezután minden elemet hozzárendeljük ahhoz a klaszterhez, amelynek reprezentáns pontjától való távolsága minimális. A besorolás után új reprezentatív pontot választunk, a klaszter középpontját. Innentől kezdve pedig addig folytatjuk a besorolást és az új középpont választást, amíg történik változás.

Formálisan, az n elemű S alaphalmazt $k \leq n$ darab $C_1, .., C_k$, páronként diszjunkt partícióra osztjuk úgy, hogy az 1.11-es képletben szereplő négyzetes hibaösszeg minimális legyen.

Mivel véletlen elemekből indultunk ki, az algoritmust még néhányszor le kell futtatnunk és azt az eredményt kell választanunk, amelyik a négyzetes hibaösszeget legjobban minimalizálja. A lépésszám így lineáris: $O(nkt)$, ahol n az elemek száma, k a keresett klaszterek száma, t pedig az iterációk száma.

k-medoid klaszterezés

Finomítása az algoritmusnak a k-medoid nevű módszer, melyben a klasztert nem a középpontja reprezentálja, hanem az adatbázisban ténylegesen előforduló, leginkább középen elhelyezkedő (a klaszter többi pontjától minimális átlagos távolságú) elem, a medoid (formálisan egy C klaszter medoidja egy olyan $m \in C$ elem, melyre

$\sum_{i \in C} d(m, i)$ minimális). Ennek előnye, hogy egyrészt kevésbé érzékeny a kívülálló pontokra, másrészt csak a távolságértékeket használja, tehát nincs semmilyen megkötés az elemekre (nem kell vektortérben lenniük). A javulás ára a költség növekedése: míg a középpont meghatározása megvan lineáris időben, a medoid megtalálása ennél költségesebb.

A gyenge pontja ezeknek az algoritmusoknak, hogy a kezdeti klaszterezés milyensége nagyban befolyásolja a végeredményüket. Az érzékenység abból fakad, hogy egy szerencsétlen kezdeti klaszterezés globális minimum helyett lokális minimumhoz vezethet. Ennek az esélyének csökkentésére a következőket tehetjük:

- Véletlenszerű választás helyett kiválaszthatjuk a két legtávolabbi elemet, majd a következő elemeket sorra úgy, hogy a már megválasztott középpontoktól való (átlagos) távolságuk legyen maximális.
- Csak az első középpontot választjuk véletlenszerűen, utána az előbbihez hasonlóan a többit úgy választjuk, hogy a már kiválasztottaktól való (átlagos) távolságuk legyen a lehető legnagyobb.

Szintén hátrány, hogy az algoritmus futásához explicit meg kell adnunk a k értéket. Ezt nem tudjuk kiküszöbölni, de különböző k-k eredményeit összevethetjük és választhatjuk azt a megoldást, amely a legjobb eredményt adta. Kezdeti beállításként a szakirodalom [6] a következő értéket javasolja:

$$k \approx \sqrt{\frac{n}{2}} \tag{1.17}$$

ahol n a klaszterezendő elemek száma.

A négyzetes hibaösszeget (1.13-as képlet) gyakran használják az algoritmusok minimalizálandó célfüggvényként, de ennek a megközelítésnek van néhány súlyos hátránya:

1. Csak elliptikus klaszterek megtalálására képes, ami azt jelenti, hogy az amorf formájú klasztereket feldarabolja kisebb kör alakúakra.
2. Rossz eredményeket ad, ha a klaszterek között nagy méretkülönbségek vannak. Nagy méretű klaszter esetén ugyanis a szélső pontok távol esnek a középponttól, ami nagy hibát eredményez, így a nagy klaszter felosztásra kerül.
3. Érzékeny az outlier pontokra, hiszen ezek az őket tartalmazó klaszter középpontját elhúzzák.

1.3.4. Sűrűség-alapú módszerek

Az eddig felsorolt klaszterező algoritmusok két közös hátrányban szenvedtek: nulladik lépésként szükség volt a klaszterek számának (k) megadására, illetve csak elliptikus alakú klasztereket tudtak azonosítani. Ezen hiányosságok kiküszöbölésére dolgozták ki a sűrűség-alapú klaszterező módszereket, melyek alapfeltevése az, hogy egy klaszteren belül jóval nagyobb az elemek sűrűsége, mint a klaszterek között.

DBSCAN klaszterezés

Az első és legismertebb sűrűség-alapú klaszterező algoritmus a DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*), melynek alapötlete az, hogy egy klasztert addig növesztünk, amíg hozzá tudunk venni olyan pontot, ami egy előre megadott ε távolságnál közelebb van valamelyik eleméhez. Úgy is fogalmazhatunk, hogy bizonyos sűrűségi feltételeket teljesítő pontok és ε-nál közelebbi szomszédaik kerülnek egy klaszterbe.

Egy p elem szomszédai ($N_\varepsilon(p)$) azok az elemek, amelyek p-től legfeljebb ε távolságra vannak. Ha $q \in N_\varepsilon(p)$ és $|N_\varepsilon(p)| \geq minPts$, akkor azt mondjuk, hogy q elem p-ből sűrűség alapon közvetlenül elérhető. Ha pedig léteznek $p = p_1, p_2, \ldots, p_n = q$ elemek úgy, hogy p_{i+1} sűrűség alapon közvetlenül elérhető p_i-ből, akkor q elem sűrűség alapon elérhető p-ből. A p és q elemek sűrűség alapon összekötöttek, ha létezik olyan o elem, amelyből p és q sűrűség alapon elérhető.

Ezek után a klasztereket a következő két szabály alapján kaphatjuk meg:

1. Ha $p \in C$ és q sűrűség alapon elérhető p-ből, akkor $q \in C$.
2. Ha $p, q \in C$, akkor p és q sűrűség alapon összekötött.

Az algoritmus első lépésében választunk egy tetszőeleges p elemet és meghatározzuk a belőle sűrűség alapon elérhető elemeket. Amennyiben $|N_\varepsilon(p)| \geq mintPts$ teljesül, kaptunk egy klasztert. A feltétel nem teljesülése nem feltétlenül jelenti azt, hogy p nem tartozik egy klaszterbe sem, lehetséges, hogy egy klaszter szélén helyezkedik el. Ilyenkor csak válasszunk egy új elemet. Ha már nem lehet új elemet választani, az algoritmus véget ért. Ha egy elem ezután sem tartozik egy klaszterbe sem, akkor zajnak tekinthetjük.

A módszer fő gyengesége, hogy rendkívül érzékeny a két paraméterre ($\varepsilon, minPts$). Sőt, ha a különböző klaszterek sűrűsége eltérő, akkor nem biztos, hogy lehet olyan ε-t és $minPts$-t megadni, ami értékelhető eredményhez vezet.

OPTICS klaszterezés

A DBSCAN egyik hátrányát küszöböli ki az OPTICS (*Ordering Points To Identify the Clustering Structure*) algoritmus: különböző sűrűségű klaszterek feltérképezésére is használható.

Alapötlete, hogy minden olyan ponthoz, amelynek legalább $minPts$ pont van ε-sugarú szomszédságában, hozzárendel egy olyan értéket is, hogy a tőle $minPts$-edik szomszédos elem milyen messze van. Ez egy lokális sűrűségi érték, melyet a későbbi lépései során arra használ, hogy ne akadályozza különböző sűrűségű klaszterek létrejöttét.

Ezt így írhatjuk le pontosan (CD = *Core Distance*):

$$CD_{\varepsilon,minPts}(p) = \begin{cases} \text{NEM DEFINIÁLT} & \text{ha } N_\varepsilon(p) < minPts \\ \text{a minPts-edik elem távolsága} & \text{egyébként} \end{cases}$$

Ekkor egy p elem o elemtől való elérhetőségi távolsága (RD = *Reachibilty Distance*):

$$RD_{\varepsilon,minPts}(p,o) = \begin{cases} \text{NEM DEFINIÁLT} & \text{ha } N_\varepsilon(o) < minPts \\ max(CD_{\varepsilon,minPts}(o), d(o,p)) & \text{egyébként} \end{cases}$$

Ezek után ez az az $\varepsilon' < \varepsilon$ érték, amely a p és o elem körüli sűrűséget számszerűsíti. Így tehát a kezdeti ε megadása csak egy maximális sűrűségi értéket határoz meg, az algoritmus nála kisebb sűrűségű klaszterek azonosítására is képes.

1.3.5. További módszerek

Megvalósításra nem került, de megemlítendő még néhány olyan technika, amely nem tartozik bele a fenti kategóriák egyikébe sem.

- Spektrális módszerek

  Ezek azok a módszerek, amelyek a klaszterezés során az adathalmaz távolságait tartalmazó mátrix sajátértékeit, illetve sajátvektorjait használják.

- Grid-alapú módszerek

  Ezek az eljárások az elemeket rácspontokba képezik le, majd a továbbiakban csak a kapott rácspontokkal dolgoznak. Fő előnyük a gyorsaságuk.

- Modell-alapú módszerek

  Az adatot valamilyen matematikai modell segítségével próbálják leírni, melyek egyben karakterizálják a beletartozó elemek tulajdonságait is. Két fő ága a döntési fa alapú megközelítés és a neurális hálókat használó eljárás.

- Fuzzy klaszterezés

  A korábbiakkal ellentétben, ahol minden elem pontosan egy osztályba tartozott, ezek az eljárások az elemek és a klaszterek között beletartozási értéket számolnak, mely annál nagyobb, minél inkább biztos, hogy az elem az adott klaszter tagja.

- Genetikus algoritmusok

  Az evolúciót utánzó genetikus algoritmusok a lehetséges megoldások egy populációját hozzák létre, amelyekhez lépésenként új egyedeket adnak, illetve a már meglévő egyedekre szelekciós, rekombinációs és mutációs operátorokat alkalmaznak [7].

2. fejezet

Az alkalmazási terület

Miután elmélyültünk az adatbányászat által nyújtott klaszterezési lehetőség és a hozzátartozó algoritmusok ismertetésében, vázoljuk fel az alkalmazási terület általános, majd a projekt szempontjából fontos jellemzőit, melyeket számításba kell majd vennünk az implementáció során.

Ki fog derülni, hogy mennyire nem újdonság a matematika és a zene összekapcsolása, és tárgyalásra kerül néhány fontos lépés a népzenekutatás történetéből is, melyek a projekt fő előzményeinek tekinthetőek.

2.1. A zene és a matematika kapcsolata

A matematika és a zene közötti szoros összefüggés már az ókori időktől kezdve ismert az emberiség számára.

A püthagoreusok mintegy 2500 évvel ezelőtt felismerték, hogy a zenei hangok világában az egész számok arányai teremtenek harmóniát [8]. Egyhúros hangszerükkel kikísérletezett, legtisztábbnak tartott hangközeikhez tartozó lefogások a kifeszített húrt $2:1$, $3:2$ és $4:3$ arányban osztják ketté, melyek rendre az oktáv, a kvint és a kvart hangközök frekvencia-arányai.

A matematika fejlődésével újabb területeket vontak be a zeneelméletbe. Leibniz a 17. században azt vizsgálta, hány különféle, 6 hangból álló dallamot lehet létrehozni oly módon, hogy a hangok ismétlődhetnek is [9]. Eulernek tulajdonítják azt a szállóigét, hogy a zene hangzó matematika. Bolyai Farkas és János munkássága is tartalmaz matematikai nyelven írt zeneelméleti értekezéseket [10].

A két absztrakció összefüggését W. R. Wade professzor a zenei érzékelésünk Fourier-sorhoz való kapcsolatával így fogalmazta meg [11]: „A klasszikus harmonikus analízis, a Fourier-analízis gyökerei mélyre nyúlnak. Mondhatnám, Isten volt az első, aki Fourier-analízist művelt, amikor fülünkbe beépített egy Fourier-analizátort. Ugyanis már a gyermek is képes arra, hogy különbséget tegyen például a hegedű és a harsona

hangja között. Annak ellenére, hogy a hangjegyek, amelyekkel a dallamot leírjuk, ugyanazok. Mi akkor a különbség a két hang között? Az, hogy amikor megszólaltatunk egy hangot, az sosem csupán tiszta hang, hanem több felhangból álló együttes. Kissé általánosabban fogalmazva: minden függvényben, ami egy hangzásnak megfelel, sok rejtett információ van, amit észlelni kell, s fülünk észlelni is tudja."

A számítógép feltalálása óta a matematika a zene még több területén szerepet kap: szintetikus hangok előállítására vagyunk képesek, algoritmusok komponálnak dallamokat, gépzenekar kíséri az élő szólistákat. A számítási kapacitás növekedésével további távlatok nyíltak a zeneelemzés területén is.

2.2. A zene számítógépes elemzése

A zene számítógépes elemzésének első mozzanatai a hatvanas évekbe nyúlnak vissza, amikor a kutatók rádöbbentek a számítógép hatalmas feldolgozóképességében rejlő kiaknázatlan lehetőségekre [12].

Az internet és az MP3 kódolás elterjedése hatalmas löketet adott a számítógépes zenei elemzésnek is. A Zenei Információ Kinyerés, vagy ismertebb nevén MIR (Music Information Retrieval) fogalma egy olyan metodológiát takar, melynek során az elemzés nem metaadatok, hanem zenei tartalom alapján történik. Ez a megközelítés tehát lehetővé tesz műfaj, vagy akár előadó, számcím meghatározását korábbi tanulófázisok alapján. Ilyen módszereket használ néhány okostelefonos alkalmazás is, amelyek a készülék mikrofonja által rögzített tetszőleges zenerészlet alapján próbálják meghatározni, hogy annak mi a címe, ki az előadója, stb...

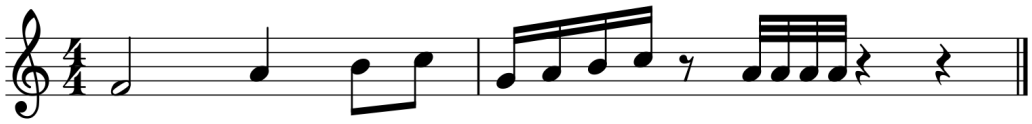
Ennek a megközelítésnek a sajátossága az előadás és a rögzítés során fellépő jelentős mennyiségű zaj. Emiatt zenekutatási szempontból érdekesebb, ha dalok modelljéül szolgáló (ezáltal zajmentes) kottákat vesszük kiindulási alapul, majd a belőlük képzett adathalmazon végzünk elemzéseket.

2.3. A zene modellje: a kotta

A kotta zenei hangok, ritmusok és a hozzájuk tartozó kiegészítő információk tárolására használt jelrendszer; a zene modellje.

A kottán a hangok párhuzamos vonalakon helyezkednek el, amelyek a hangmagasságokat jelölik. A zenei hangoknak gombócok felelnek meg, melyeket a latin ábécé (C, D, E, F, G, A, H) betűiről neveztek el. Ha egy hang mélysége vagy magassága miatt nem fér el az alap öt vonal valamelyikén (vagy közöttük), a kotta pótvonalakkal egészíthető ki.

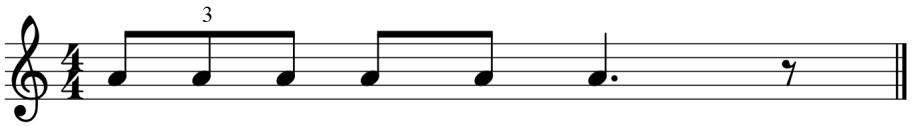
A violinkulcs után álló két egymás fölött lévő egész szám azt jelöli, hogy az ütemek hány azonos hosszúságú hangjegyet tartalmazhatnak. Az ütemeket függőleges



2.1. ábra. *Egyszerű kotta fél-, negyed-, nyolcad-, tizenhatod-, és harmincketted hangokkal, valamint szünetekkel*

vonal, az ütemvonal választja el egymástól. Az ezek közti hangok (és szünetek) összhosszúsága azonos, és megegyezik az ütemjelzésben megadott értékkel (4/4 esetén négy negyed hosszú ütemek).

A zene ritmusát a hangok hossza határozza meg. Az egységhosszú hangot üres, szár nélküli gombóc jelöli, a félhangot üres gombóc szárral, a negyedet (tá) teli gombóc szárral. A negyednél kisebb hangok esetén a hangokat összekötő gerendák száma határozza meg a hang hosszúságát. Egy gerendával vannak összekötve a nyolcadok (ti), kettővel a tizenhatodok, hárommal a harminckettedek és így tovább (Lásd: 2.1 ábra). Az aktuálisan jelölt hang hosszát másfélszerezi a gombóc melletti pont, így egy negyedből három nyolcad, nyolcadból három tizenhatod, stb. . . hosszúságú hangot képez.



2.2. ábra. *Egyszerű kotta triolával és pontozott hanggal*

A triola egy olyan három hangjegyből álló csoport, amely nem rendes értékekre támaszkodik. Például ha szabályszerűleg egy negyed hangjegyre két nyolcad esne, triola esetén a hangot három „nyolcadra” osztjuk. A kottán ezt egy szám jelöli a hangokat összekötő gerendán (Lásd: 2.2 ábra). Öt hangjegyből álló verziójának neve kvintola, ekkor négy tizenhatod helyett öt egyenlő részre osztjuk a negyedhangot.

Ezeken kívül vannak további módosító-, illetve feloldójelek, melyek részletezése megtalálható a szakirodalomban [13].

2.4. A népzenekutatás

A népzenekutatás az összehasonlító zenetudományból alakult ki a 19. század végén [14]. Ekkoriban vált lehetővé a dalok nagyszámú pontos rögzítése a fonográf segítségével, ami a későbbi tudományos lejegyzést és rendszerezést is elősegítette.

2.4.1. A zenei rendszerezés

A fonográfos néprajzi gyűjtések megindulása és sikere után hamarosan létrejöttek a nemzeti gyűjtemények. A Néprajzi Múzeum 4,500 második világháború előtti fonográfhenger-gyűjteménye a világ legrégebbi és leggazdagabb készleteinek egyike [14]. Az „adatbázis" növekedésével együtt megjelent az igény a gyűjtések rendszerezésére is.

A rendezésre törekvő elképzeléseknek mindenekelőtt az alábbi két szempontot kellett követniük:

- szótárszerű, könnyen kezelhető legyen, hogy bárki megérthesse - így a dalok könnyen kikereshetőek
- ügyeljen arra, hogy a variánsok, a rokon típusok egymás mellett, vagy legalábbis egymás közelében legyenek - így kaphatunk pontos és tiszta képet egy-egy vidék népzenéjéről

A továbbiakban sorra veszünk néhány példát a népzene tudományos rendszerezésének történetéből.

Bartók Béla rendje

Hazánkban a népdalokról szóló első összefoglaló munka Bartók Béla nevéhez fűződik [15], aki formai alapon a következő három nagy stíluscsoportba osztotta a népdalokat:

- Régi stílus
- Új stílus
- Vegyes stílus

Ennek a csoportosításnak a fő szempontja nem a szöveg, hanem a magyar, illetve az idegen eredet. Az 1. és 2. csoport a magyar dallamfejlődést tükrözi, míg a 3. csoport az idegen dallamokat. A csoportokon belül a dalok főként ritmusbéli sajátosságok alapján rendeződnek.

Kodály Zoltán rendje

Kodály Zoltán A magyar népzene című népzene-történeti összefoglalása 1937-ben jelent meg [16]. Ezzel a kötettel honosodott meg Magyarországon az összehasonlító zenetudomány.

A tanulmányban a dallamok tulajdonságait tartotta a fő szempontnak, egészen pontosan a sorvégi kadencia (záró) hangok adták az elsődleges rendező elvet. Tovább haladva a sorok szótagszáma alapján rendeződnek a dalok.

Népzenei típusrend

Kezdetektől fogva megvolt az igény arra, hogy a hasonló dallamok egymás mellé kerüljenek a rendszerezés során. Járdányi Pál nevéhez fűződik az első, teljes dallamvonalak hasonlóságát figyelembe vevő rendszer. Dobszay László és Szendrei Janka dolgozta ki a ma használatos típusrendszert, mely a Zenetudományi Intézet 200,000 körüli dallamból álló készletét olyan hangkészlet-alapú csoportokra bontja, melyek a leghűbben reprezentálják a magyar népzene egészét [8]. Ilyenek például:

- Kvintváltó és egyéb pentaton, vagy pentatonból leszármaztatható ereszkedő dallamok
- Dúr, oktávambitusú ereszkedő dallamok
- Pszalmodizáló stílus
- Sirató stílus
- Dudanóta-kanásztánc stílus
- Új stílus
- Stb...

2.4.2. A magyar népzene kapcsolatai

A magyar népzene első összehasonlító népzenei elemzése is Bartók és Kodály nevéhez fűződik.

A bartóki típusú régi stílusú dallamok a belső-ázsiai türk népek zenéjével rokonok [17], illetve az anatóliai törökökével is találtak hasonlóságokat [18]. Érdekes módon a finnugor népek népzenéje nem áll kapcsolatban a régi stílusú népdalosztállyal [14]. A Kárpát-medencei, sőt, az európai népzene sem mutat rokonságot a régi stílusú magyar dalokkal.

Ellentétben a régi stílussal, az új stílusú magyar népzenéből nagyon sokat kölcsönöztek a környező népek.

3. fejezet

A ritmus

A ritmus görög eredetű szó, jelentése: lüktetés, dobogás, (szív-)verés, ütem. Hasonló vagy azonos elemek időbeli szabályos váltakozását, ismétlődését értjük alatta. A zene csak az időben fogható fel, tehát nem értelmezhető a ritmus fogalma nélkül.

A dallam és a ritmus együttesen határoznak meg egy dalt, azt azonban nem mondhatjuk, hogy egymástól elválaszthatatlanok, hiszen gyakran ugyanazt a dallamot teljesen más ritmussal hallhatjuk, illetve a különböző eredetű és tulajdonságú dallamok előfordulnak azonos ritmussal [19].

A népdalok rendszerezésében a dallam és a ritmus szerinti megközelítés mindig párhuzamosan érvényesült.

3.1. A ritmusok összehasonlítása

Mind adatbányászati, mind zeneelméleti szempontból alapvető kérdés, hogy hogyan hasonlítsunk össze két ritmust. A ritmusok reprezentációját úgy kell megválasztanunk, hogy azok összehasonlíthatóak legyenek olyan módon, amit a zenekutató értelmesnek talál. Alapvető követelmény, hogy ez az összehasonlítás emberi észlelésünkkel összhangban legyen, azaz hasonlónak érzékelt ritmusokat hasonlónak, különbözőnek érzékelt ritmusokat pedig különbözőnek találjon.

Két ritmus (r_i és r_j) különbözőségét valamiképp mérhetővé kell tennünk. Meg kell adnunk egy formulát ($d(r_i, r_j)$), amely a kapott két ritmusábrázolás alapján képes meghatározni egy különbözőséget tükröző numerikus értéket. Kézenfekvő megközelítésnek tűnik, ha ezt a különbözőségi értéket a 0 és 1 közötti skálán helyezzük el, ahol minél kisebb az érték, annál jobban hasonlít egymásra a két ritmus. A 0 jelenti a teljes ritmikai azonosságot, míg az 1 a lehető legnagyobb különbözőséget, a két szélsőérték között pedig egyenesen arányosan oszlik el a különbözőség.

Az általános távolságfüggvénnyel szemben támasztott formális követelmények a következők:

1. $d(x,y) \geq 0, \forall x,y$-ra és $d(x,y) = 0 \Leftrightarrow x = y$, azaz két elem távolsága nem negatív, és nulla akkor és csak akkor, ha a két elem azonos.

2. $d(x,y) = d(y,x)$. Az x és az y elem távolsága mindkét irányban ugyanaz (szimmetria).

3. $d(x,z) \leq d(x,y) + d(y,z), \forall x,y,z$-re (háromszög-egyenlőtlenség).

A szimmetria megkövetelése esetünkben nem feltétlenül szükséges, ezt a feladat speciális jellemzői és a kiválasztott metrika sajátosságai határozzák meg.

3.2. A távolságok tárolása

Az alapelképzelés szerint minden ritmus összehasonlításra kerül az összes többi ritmussal, a kiszámított értékek pedig egy úgynevezett különbözőségi mátrix (M_{diff}) megfelelő helyére kerülnek. Ezt úgy képezzük, hogy vesszük egy sorba az összes ritmust, illetve vesszük őket egy oszlopba is, majd a kapott táblázat megfelelő helyére beírjuk a különbözőségi értéket. A mátrix főátlójában nyilvánvalóan csupa nullák állnak, hiszen r_i ritmus megegyezik r_i ritmussal, azaz $d(r_i, r_i) = 0, \forall i$-re (3.1 képlet). Ebben az esetben $n(n-1)$ összehasonlítást kell elvégeznünk és ennyi eredményt is kell tárolnunk.

$$M_{diff} = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{n-1} \\ r_n \end{array} \begin{array}{c} \begin{array}{cccccc} r_1 & r_2 & r_3 & \cdots & r_{n-1} & r_n \end{array} \\ \begin{pmatrix} 0 & d(r_1,r_2) & d(r_1,r_3) & \cdots & d(r_1,r_{n-1}) & d(r_1,r_n) \\ d(r_2,r_1) & 0 & d(r_2,r_3) & \cdots & d(r_2,r_{n-1}) & d(r_2,r_n) \\ d(r_3,r_1) & d(r_3,r_2) & 0 & \cdots & d(r_3,r_{n-1}) & d(r_3,r_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ d(r_{n-1},r_1) & d(r_{n-1},r_2) & d(r_{n-1},r_3) & \cdots & 0 & d(r_{n-1},r_n) \\ d(r_n,r_1) & d(r_n,r_2) & d(r_n,r_3) & \cdots & d(r_n,r_{n-1}) & 0 \end{pmatrix} \end{array} \tag{3.1}$$

Amennyiben úgy döntünk, hogy a megköveteljük távolságfüggvény szimmetriáját is, akkor a mátrix is szimmetrikus lesz (hiszen ekkor lényegtelen, hogy r_i ritmust vetjük össze r_j-vel, vagy r_j ritmust r_i-val, azaz $d(r_i, r_j) = d(r_j, r_i), \forall i,j$-re). Ekkor a könnyebb tárolás és feldolgozás érdekében elég, ha csak a főátló alatti elemeket tekintjük.

$$M_{diff} = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{n-1} \\ r_n \end{array} \begin{array}{c} \begin{array}{cccccc} r_1 & r_2 & r_3 & \cdots & r_{n-1} & r_n \end{array} \\ \begin{pmatrix} 0 & - & - & \cdots & - & - \\ d(r_2, r_1) & 0 & - & \cdots & - & - \\ d(r_3, r_1) & d(r_3, r_2) & 0 & \cdots & - & - \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ d(r_{n-1}, r_1) & d(r_{n-1}, r_2) & d(r_{n-1}, r_3) & \cdots & 0 & - \\ d(r_n, r_1) & d(r_n, r_2) & d(r_n, r_3) & \cdots & d(r_n, r_{n-1}) & 0 \end{pmatrix} \end{array} \tag{3.2}$$

A 3.2 képletben szereplő mátrix-definícióban látható, hogy szimmetria esetén mely értékekre nincs szükségünk, melyek adottak és melyeket kell kiszámolnunk. Adott n számú ritmust feltételezve könnyű meghatározni, hogy ehhez összesen $\frac{n(n-1)}{2}$ összehasonlítás szükséges.

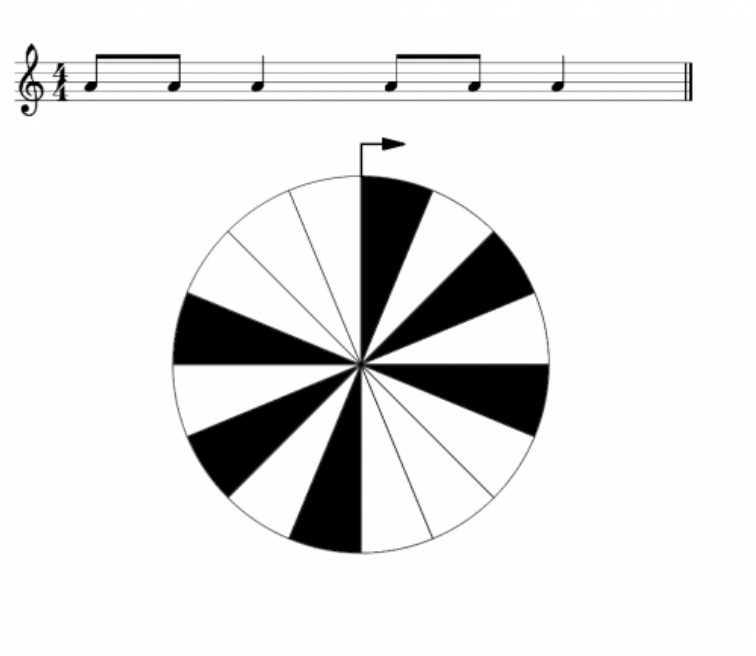
Néhány tízezer elemből álló adatbázis esetén az így kapott M_{diff} memóriában is tárolható, ha azonban továbbnő az adathalmaz mérete, megfontolandó a memóriában csak minden elem k legközelebbi szomszédjától való távolságát tárolni. A projekt jelenlegi fázisában a kisebb méret miatt ezzel nem kell foglalkoznunk.

3.3. A ritmus geometriai reprezentációja

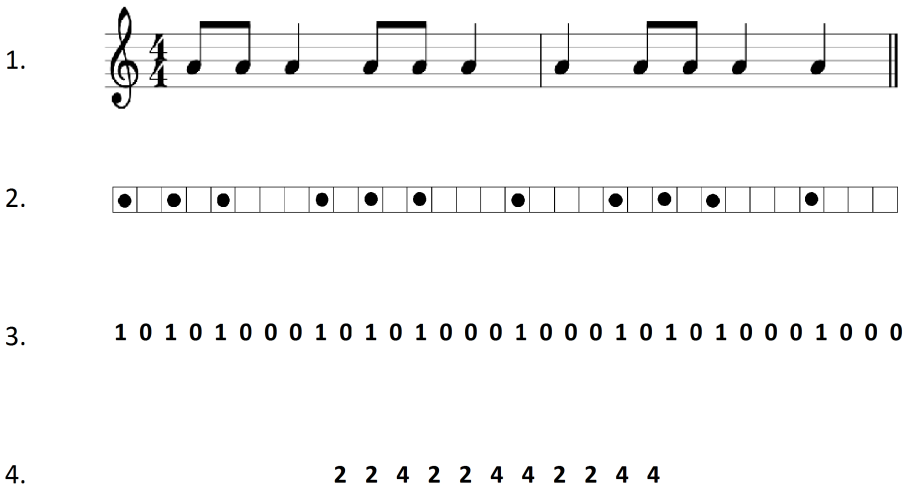
A ritmus vizuális megjelenítése nem csak könnyebb elemzést tesz lehetővé, de olyan tulajdonságokra hívhatja fel a figyelmet, melyek fontosak a hasonlóság méréséhez.

A 13. századi arab zenetudós, Safi al-Din óta ismert a ritmus azon szemléltetése, mely egy kört körcikkekre bont, az ütésekhez tartozó körcikkeket kiszínezi, a szünetekhez tartozókat pedig üresen hagyja [20]. A kezdőpontot egy nyíl jelzi. Ez a zseniálisan egyszerű módszer a ritmus ciklikusságát is megragadja, hiszen egy ritmus több versszakból, strófából álló dalok esetén visszatér.

A 3.1 ábrán látható egy egyszerű ti-ti-tá ti-ti-tá ritmus ábrázolása. A teljes kör 16 egyenlő körcikkre van osztva, a hangsúlyos ütemek elejéhez tartozó körcikkek kitöltöttek, az utánuk következő, az ütem hosszúságától függő számú körcikkek pedig kitöltetlenek. Rövidebb ütemek (pl. tizenhatodok) szemléletes ábrázolásához nagyobb felbontású köröket kell alkalmaznunk. Az ábra tengelyes vagy középpontos szimmetriája fontos információt hordoz a megjelenített ritmusról is.



3.1. ábra. *A ritmus 13. századból származó ciklikus ábrázolása*

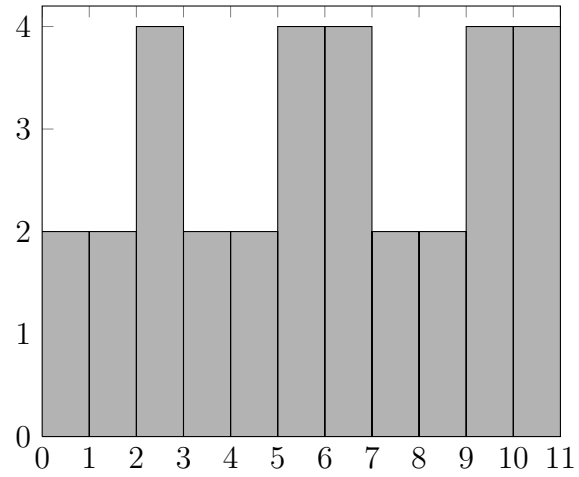


3.2. ábra. *Négy különböző ritmusábrázolás*

3.3.1. Time Unit Box System

A ritmusok gépi ábrázolására a University of California (UCLA) kutatói által kifejlesztett Time Unit Box System (TUBS) módszer tűnik a legalkalmasabbnak (a 3.2 ábrán a 2. módszer). Ez néhány egymás mellé helyezett dobozból áll, melyek mindegyike egy előre meghatározott időszeletet reprezentál. A dobozok vagy üresek, vagy jelöltek lehetnek, az üres doboz a hozzá tartozó időszeletben zenei esemény

hiányát, míg a jelölt doboz a annak meglétét tükrözi. Ez a megközelítés könnyedén interpretálható a számítástudomány által egyszerűen feldolgozható bináris vektorok nyelvén is (a 3.2 ábrán a 3. módszer).



3.3. ábra. *Szomszédos intervallumok spektruma*

Megjegyzendő, hogy triolák, illetve egyéb nem szabályos értékekre támaszkodó ritmikai elemek reprezentálására ez a módszer csak jelentős felbontás növekedéssel alkalmas. A felbontás pontos mértéke függ az előforduló szabálytalan elemek típusától is.

3.3.2. Szomszédos intervallumok ábrázolása

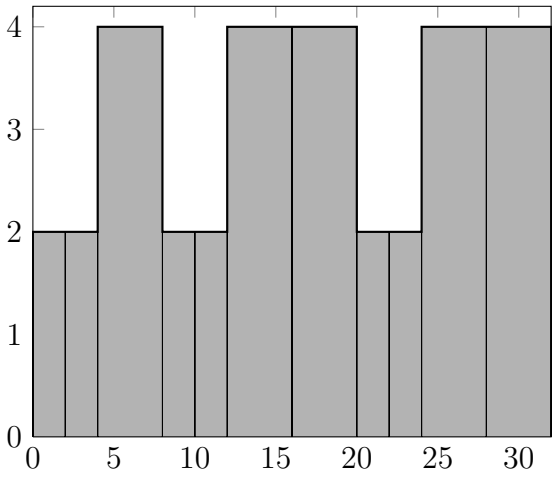
A 3.2 ábrán látható 4. ritmusábrázolási módszerben található számok azt jelzik, hogy milyen hosszúak az egymást követő hangsúlyos elemek. A ti-ti-tá ritmus így tizenhatodos felbontással a

$$\begin{pmatrix} 2 & 2 & 4 \end{pmatrix}$$

intervallum-vektorral jellemezhető.

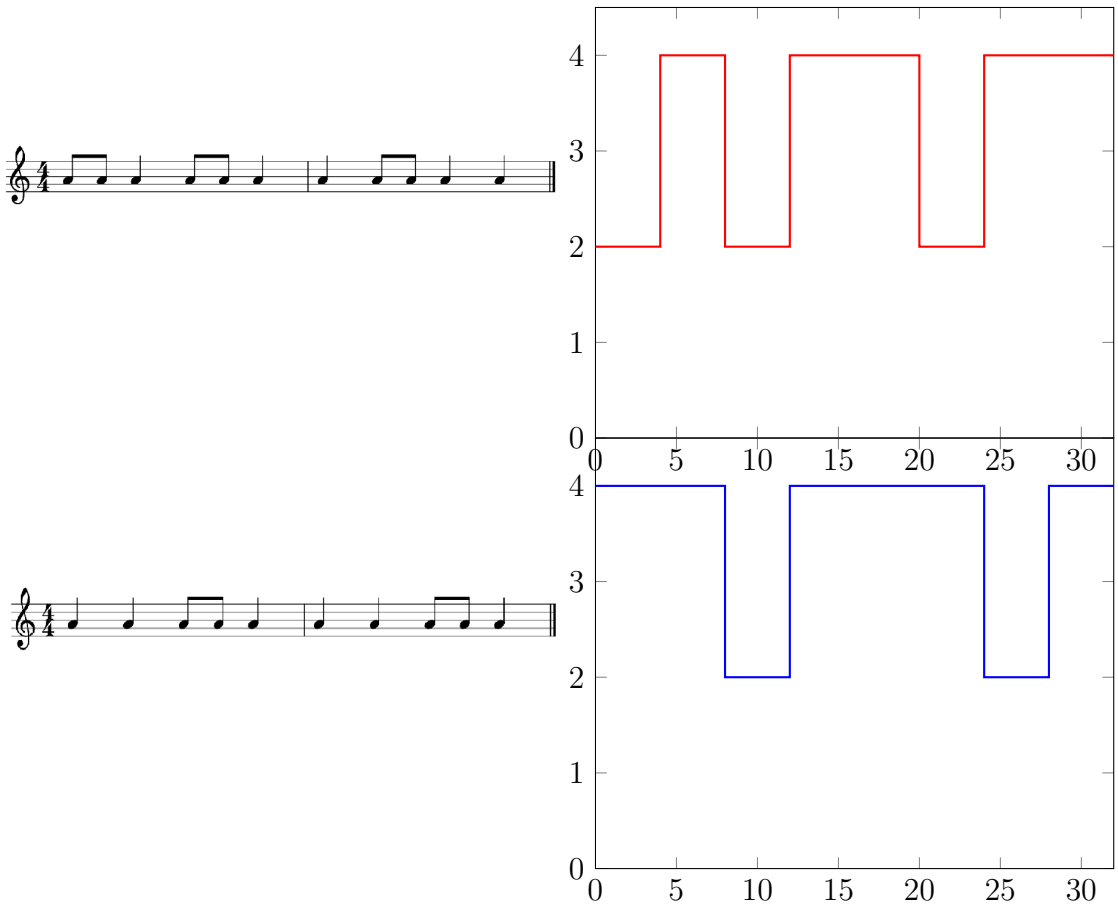
Ennek a vektornak a szemléltetésére a University of Oxford kutatói dolgoztak ki módszereket. Ezek egyike, a szomszédos intervallumok spektrumának módszere úgy ábrázolja a ritmust egy koordináta-rendszerben, hogy az x tengely mentén egyesével haladva egy-egy oszlop magassága felel meg az adott pozícióban álló hangsúly hosszának (3.3 ábra). Az azonos ütemet tartalmazó ritmusok így szemléletesen ábrázolhatóak egymás mellett vagy alatt, de ez a módszer megnehezíti az intervallumok relatív hosszúságának érzékelését.

Kézenfekvő ötlet a hátrányok kiküszöbölésére a különböző módszerek ötvözete. A TUBS módszert és a szomszédos intervallumok spektrumának módszerét úgy tudjuk vegyíteni, hogy minden TUBS-ban szereplő dobozba azt a számot írjuk, amilyen hosszú a hozzá tartozó vagy az őt megelőző hangsúlyos elem. Ha az így kapott vek-



3.4. ábra. *Temporal Elements Displayed As Squares (TEDAS)*

tort ábrázoljuk a koordináta-rendszerben, akkor egy olyan ábrát kapunk, melyen az időbeli elemek megfelelő méretű négyszögekkel vannak reprezentálva. Ebből származik a módszer elnevezése is: Temporal Elements Displayed As Squares, vagy röviden TEDAS [21].



(a) *Ritmusok* **(b)** *Kronotonikus láncok*

3.5. ábra. *Kronotonikus láncok értelmezése*

A korábbiakban tárgyalt ritmushoz tartozó TEDAS ábrázolás látható a 3.4 ábrán. Az alapjául szolgáló vektort, melyet kronotonikus láncnak nevezünk, nem csak oszlopként, hanem függvényként is ábrázolhatjuk a koordináta-rendszerben (3.5 ábra).

3.4. Ritmikus különbözőségi metrikák

A mindennapi életben hosszúságként értelmezett távolság fogalmát a matematika általánosítja és különböző módokon definiált mértékeket, metrikákat vezet be. A ritmus-reprezentációkat akkor tudjuk összehasonlíthatóvá tenni, ha a köztük lévő távolságot egy nemnegatív skalármennyiséggel jelöljük. Az irodalom bőségesen foglalkozik a témával, az általam választott módszerek nagyobbik része Godfried Toussaint [22], illetve Lior Rokach [4] cikkeiből származik, de gyakran további pontosításra és kiegészítésre szolgáltak az általuk vázolt általánosabb eljárások.

3.4.1. Hamming-távolság

A számítástudományban bináris vektorok összehasonlításának legtermészetesebb módja a vektorok elemenkénti összehasonlítása, ismertebb nevén Hamming-távolság számolása. Ebben a megközelítésben a TUBS módon reprezentált ritmusok közvetlenül használhatók, hiszen azok lényegében bináris vektorok. A módszer elméleti hátterében az n hosszú vektorok egy n-dimenziós hiperkocka pontjai. Adott $X = (x_1, x_2, ..., x_n)$ és $Y = (y_1, y_2, ..., y_n)$ közötti távolság számolható az alábbi képlet segítségével:

$$d_H(X, Y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.3}$$

ahol $|x|$ az x érték abszolút értékét jelöli.

A módszer nagyon egyszerű, időigénye is alacsony, hiszen elég egyetlen ciklusban végigmenni a vektorokon és akkumulálni a különböző bitek számát, azaz lépésszáma a vektor hosszának konstans-szorosa ($O(n)$). Ugyanakkor nem biztos, hogy erre a speciális igényre a legjobban megfelel, mert az különbözőséget ugyan jól méri, de azt nem, hogy milyen messze vannak egymástól az eltérések. A gyakorlati tapasztalatok azt mutatják, hogy egy hangsúly távolabbra helyezése különbözőbbnek hat, mintha közelebbre tettük volna át. Szintén hátránya a módszernek, hogy egy ütemmel eltolva ugyanaz a ritmus teljesen másnak értékelődik, holott az emberi fül számára az nagyon hasonló az eredetihez.

3.4.2. Bináris vektorok távolsága

Az irodalom [4] által javasolt általános módszer bináris vektorok összehasonlítására két esetet különböztet meg. Abban az esetben, ha a bináris vektorban a 0 illetve az 1 érték azonos valószínűséggel fordul elő, szimmetrikus attribútumról beszélünk és a következőképp határozhatjuk meg az X és Y bináris vektorok közötti távolságot:

$$d_{Bin1}(X,Y) = \frac{r+s}{q+r+s+t} \tag{3.4}$$

ahol q az azonos helyen lévő 1-esek, t az azonos helyen lévő 0-ák, s és r pedig a különböző bitek száma X-ben, illetve Y-ban (ha azonos hosszúak, akkor $s = r$). Ez egyfajta súlyozott Hamming-távolságként is értelmezhető.

A korábbiakkal szemben aszimmetrikus attribútumról akkor beszélünk, ha a 0 és 1 különböző valószínűséggel fordul elő a vektorokban, így tehát az egyikük egyezése „fontosabb", mint a másiké (általában ez az 1-esek egyezése). Ekkor a nevezőből elhagyjuk a nullák egyezését számláló t-t:

$$d_{Bin2}(X,Y) = \frac{r+s}{q+r+s} \tag{3.5}$$

A ritmusokat reprezentáló TUBS vektorok általános jellemzője, hogy több bennük a szünet, mint a hangsúlyos elem (az 1-es), ezért a másodikként definiált megközelítést használjuk inkább.

Ez a megközelítés sajnos szintén rendelkezik a Hamming-távolság minden, korábban vázolt hátrányával.

3.4.3. Intervallum-vektorok Manhattan-távolsága

Zenefelismerő és zenekategorizáló algoritmusok által gyakrabban használt ritmusábrázolás a hangsúlyok közötti intervallumok hosszait reprezentáló vektorokon alapul (a 3.2 ábrán a 4. módszer hasonló, de ott a hangsúlyos ütemet is beleszámoljuk a hosszba, ezért ott a vektor minden eleme eggyel nagyobb). Egy ritmus az $X = (x_1, x_2, ..., x_n)$ vektor által reprezentált, ahol x_1 jelöli az első hangsúly utáni intervallum hosszát. Az $X = (x_1, x_2, ..., x_n)$ és $Y = (y_1, y_2, ..., y_n)$ által jelölt ritmusok közötti Manhattan-távolságot az alábbi formula adja meg:

$$d_M(X,Y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.6}$$

Az előző módszertől mindössze annyiban tér el, hogy egy más módon képzett vektort vesz kiindulási alapul, melynek elemei nem csak a $\{0,1\}$ halmazból kerülhetnek ki. Az intervallum-vektorok $O(n)$ időben számolhatók a TUBS ábrázolásból, ebből

következően a lépésszámban nincs eltérés ($O(n)$). Ez a megközelítés úgy értelmezhető a mindennapi távolsághoz képest, hogy egy olyan út hosszát méri, mely egymásra merőleges szakaszokon haladva vezet az egyik pontból a másikba, mintha csak egy nagyvárosi úthálózaton haladhatnánk.

3.4.4. Intervallum-vektorok Euklideszi-távolsága

Az előző módszert továbbgondolva könnyen eljuthatunk a mindennapi értelemben vett távolsághoz legkézenfekvőbben illeszkedő módszerhez, az Euklideszi-távolsághoz. Egy, két, illetve három dimenzióban pontosan azt is adja eredményül, mint amit egy vonalzóval mérve kapnánk. Egy ritmus ugyanaz az $X = (x_1, x_2, ..., x_n)$ vektor által reprezentált, mint az előző szakaszban. Az $X = (x_1, x_2, ..., x_n)$ és $Y = (y_1, y_2, ..., y_n)$ által jelölt ritmusok közötti Euklideszi távolságot így adhatjuk meg:

$$d_E(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.7}$$

A korábbiakhoz hasonlóan ez az algoritmus is $O(n)$ lépést igényel.

Az utóbbi két megközelítés a Hamming-távolságnál alkalmasabb a ritmus különbözőségének meghatározásához, hiszen az eltolási problémát kiküszöbölik.

3.4.5. Az intervallum-különbözőségi vektorok távolsága

Ez a módszer Coyle-tól és Shmulevich-től származik. Az előzőekhez hasonlóan ez a megközelítés is az intervallum-vektorokból indul ki, ám mielőtt a két ritmus intervallumhosszait egymáshoz hasonlítaná, az azonos ritmusban szomszédos elemekhez méri őket. Így egy ritmus az egymást követő intervallumok arányival van jellemezve. Formálisan a $T = (t_1, t_2, ..., t_n)$ intervallum-vektor által jelölt ritmushoz egy olyan $X = (x_1, x_2, ..., x_n)$ vektort definiál, melynek elemei $x_i = t_{i+1}/t_i$ módon vannak meghatározva. A ciklikusságot is figyelembe véve $x_n = t_1/t_n$, így T-hez hasonlóan X dimenziója is n lesz. Az így kapott X vektort intervallum-különbözőségi vektornak nevezzük. Az $X = (x_1, x_2, ..., x_n)$ és $Y = (y_1, y_2, ..., y_n)$ intervallum-különbözőségi vektorok által jelölt ritmusok közötti távolságot így definiáljuk:

$$d_{ID}(X, Y) = \Big(\sum_{i=1}^{n} \frac{max(x_i, y_i)}{min(x_i, y_i)}\Big) - n \tag{3.8}$$

A szummázandó hányados értéke mindig legalább 1, ezért kell levonnunk belőle a végén n-t, hiszen annak nincs információtartalma. Az algoritmus a korábbiakhoz hasonlóan $O(n)$ időben fut.

3.4.6. Cserélési távolság

A negyedik, korábbiaktól teljesen eltérő megközelítés nem a különbözőségeket keresi, hanem azt számolja ki, hogy mennyi munka szükséges az egyik vektort a másikba transzformálni. A genetikában széles körben használt módszer molekulák egymásba alakításához szükséges alaplépések számát határozza meg, ahol az alaplépések valamilyen evolúciós mutációt szimulálnak. Nekünk egyszerűbb dolgunk van, az alaplépés ugyanis a pofonegyszerű csere művelete. Annyit azonban meg kell kötnünk, hogy csak szomszédos elemeket cserélhetünk ki egymással.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Például, a fenti ritmusvektor 4 lépésben alakítható át az

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

vektorrá, mégpedig a harmadik, az ötödik és a hetedik hangsúlyos elem és az őket követő hangsúlytalan elem cseréjével.

A cserélési távolság kiszámításához azonban nincs szükségünk a tényleges cserék végrehajtására, hiszen az bizonyos esetekben nagyon költséges lenne. Ilyen például az alábbi két ritmusvektor:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

ahol a ritmus hosszának négyzetével arányos csere szükséges. Ehelyett egy sokkal elegánsabb megoldást mutatunk. Előbb eltároljuk azokat a pozíciókat, ahol hangsúlyos elem van. A fejezet első példáira ez $U = (u_1, u_2, ..., u_n) = (1, 3, 5, 6, 8, 10, 12)$ és $V = (v_1, v_2, ..., v_n) = (1, 3, 4, 6, 7, 9, 11)$. Az u_i és v_i értékek különbsége határozza meg, hogy mennyi csere szükséges mindenképpen ahhoz, hogy a két értékhez tartozó hangsúlyos elem a megfelelő helyre kerüljön. Általánosságban tehát a cserélési távolság a következőképp számítható a fenti U és V vektorok alapján:

$$d_{SWAP}(U, V) = \sum_{i=1}^{n} |u_i - v_i| \tag{3.9}$$

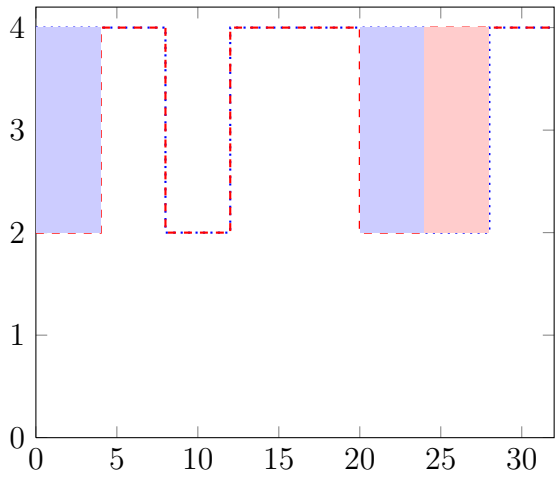
Az U és V vektorok számolása TUBS vektorokból $O(n)$ idejű, így $d(U, V)$ meghatározásához sem kell több idő.

3.4.7. Kronotonikus távolság

A Hofmann-Engl által javasolt módszer a közös koordináta-rendszerben ábrázolt kronotonikus láncokból indul ki. Két ritmus távolságát úgy határozza meg, hogy az általuk meghatározott kronotonikus láncok alatti területeket hasonlítja össze. Formálisan tehát előbb vesszük az $f_1(x)$ és $f_2(x)$ függvényekkel leírt kronotonikus láncok különbségének abszolút értékét (amire azért van szükség, hogy ne nullázhassa ki két azonos méretű eltérés egymást), majd ennek integrálját (azaz a függvény alatti terület nagyságát) tekintjük a távolságnak.

$$d_K(X,Y) = \int |f_1(x) - f_2(x)|dx \tag{3.10}$$

A 3.6 ábra szemlélteti a korábbiakból már ismert ritmusokhoz tartozó kronotonikus láncok közötti kronotonikus távolságot.



3.6. ábra. *Két ritmus közötti kronotonikus távolság*

3.5. Különböző hosszúságú ritmusok

A gyakorlatban a legritkább esetben fordul elő, hogy minden vizsgált ritmus hossza azonos. Az eddig tárgyalt módszerek azonban egytől egyig feltételezték ezt. Valamiképpen tehát módosítanunk kell a kiindulási algoritmusokon, hogy figyelembe vegyék a ritmushosszok különbözőségét.

Ugyanakkor ez a probléma ismét felveti azt a kérdést is, hogy a definiált távolságtól megköveteljük-e a szimmetriát, azaz $d(x,y)$ és $d(y,x)$ legyen-e egyenlő $\forall x, y$-ra, hiszen különböző hosszúság esetén nem mindegy, hogy milyen sorrendben hajtunk végre bizonyos lépéseket.

Szimmetria teljesülése esetén

Előbb vizsgáljuk meg azt az esetet, amikor teljesül a szimmetria. Ez esetben kézenfekvő megközelítés, ha a hosszabb ritmusvektort fixen hagyva tologatjuk („shift-eljük") a rövidebb vektort a hosszabb elejétől a végéig, minden lehetséges módon számítunk egy távolságot, majd a kapott értékek minimumát vesszük.

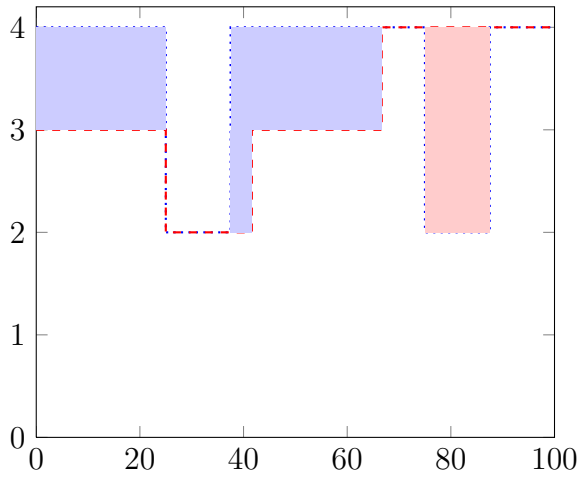
Formálisan tehát, ha $|X| = n_1$, $|Y| = n_2$ és $n_1 > n_2$, akkor

$$d_{Shifted}(X, Y) = \min_{i \in \{0,\ldots,n_1-n_2\}} d_{Spec}(X_{i,n_2+i}, Y) \tag{3.11}$$

ahol X_{i,n_2+i} jelöli X vektor i. és n_2+i. közötti elemei által alkotott vektort, beleértve a határokat is. Az ily módon meghatározott vektorok hossza már azonos, így átadhatók a korábbi szakaszban definiált eljárásoknak ($d_{Spec} \in \{d_H, d_M, d_E, d_{ID}, d_{SWAP}\}$).

Ez a módszer szinkronban van azzal a gyakorlati megfigyeléssel, hogy ha egy ritmus valahol magában foglal egy másikat, akkor szoros kapcsolat van közöttük. A megközelítés hátránya azonban, hogy nagy különbségek esetén jelentősen megnövelheti a futásidőt ($\forall i \in \{0, \ldots, n_1 - n_2\}$-re le kell futtatni az eredeti algoritmust).

A görbe alatti területekkel számoló kronotonikus távolság alapú algoritmusnál ennél egy szofisztikáltabb megközelítést alkalmazhatunk. A módszer lényege, hogy a ritmusvektorokhoz rendelt kronotonikus vektorokat egy adott hosszúságú x-tengelyre vetítjük, függetlenül azok tényleges hosszától. A különbségek meghatározása és az integrálszámítás az így kapott függvényeken történik.



3.7. ábra. *Különböző hosszúságú ritmusok közötti kronotonikus távolság 100 egység hosszú x tengelyre vetítve*

Szimmetria nem teljesülése esetén

Ha úgy döntünk, hogy a nem kívánjuk megkövetelni a szimmetria teljesülését, akkor rendelhetünk különböző értékeket a rövidebb hosszabbtól, illetve a hosszabb

rövidebbtől való távolságértékeként.

Talán nem tévedünk nagyot, ha egy rövidebb vektorra, melyet magában foglal egy másik, azt mondjuk, hogy 0 távolságra van a hosszabbtól. A hosszabb távolsága a rövidebbtől pedig annak függvényében közelít a 0-hoz, hogy mekkora a közöttük lévő hosszeltérés.

A nem szimmetrikus távolságra nézzünk egy esetet, amelynél számít a sorrend $d_H(x, y)$ paraméterezésében (nevezzük aszimmetrikus Hamming-távolságnak).

A módszer során nagyjából az történik, hogy a szummázás előtt a második vektort, amennyiben az rövidebb, mint az első, nullákkal egészítjük ki, hogy azonos hosszúak legyenek. Ezek után ugyanúgy vizsgáljuk meg a kapott vektorok távolságát, mint az eredeti módszernél.

Formálisan, adott $X = (x_1, x_2, ..., x_n)$ a nála rövidebb $Y = (y_1, y_2, ..., y_m)$-től való távolsága (tehát $|X| = n$, $|Y| = m$, és $n > m$):

$$d_H(X, Y) = \sum_{i=1}^{m} |x_i - y_i| + \sum_{i=m+1}^{n} |x_i| \tag{3.12}$$

hiszen a második tagban 0-kat vonunk le az aktuális x_i-ből.

Ezzel a módszerrel azt a valós megfigyelést realizálhatjuk, hogy ha egy X vektor része egy hosszabb Y-nak, akkor $d(X, Y)$ valóban 0, ezzel szemben $d(Y, X)$ nem feltétlenül egyenlő vele. Ha pedig adódik egy X-nél hosszabb, Y-nál még mindig rövidebb Z vektor, mely szintúgy része Y-nak, akkor $d(Y, Z) \leq d(Y, X)$.

Az eljárás ugyanígy értelmezhető Manhattan-, és Euklideszi-távolságokra, de sajnos nem alkalmazható intervallum-különbözőségi vektorok, cserélési távolság és kronotonikus láncok esetén.

4. fejezet

A megvalósítás

Az előző fejezetekben definiáltam egy problémát számos megoldási lehetőséggel. A feladatom az volt, hogy hozzak létre egy szoftvert, amely erre a problémára választ ad. Ehhez implementálni kellett a kották beolvasását, a ritmus kinyerését, a távolságok meghatározását, a klaszterezés futtatását és az eredmények megjelenítését.

A szoftver elkészítéséhez a Java nyelvet választottam annak platformfüggetlensége, széleskörű elterjedtsége, támogatottsága és rendelkezésre álló funkcionalitásának sokszínűsége miatt.

A kód, a dokumentáció, illetve a grafikus elemek szövegei is mind angolul készültek.

4.1. Forrásfájlok feldolgozása

A vizsgálandó dalok nem hangfájlként állnak rendelkezésre, hanem speciálisan kódolt kották formájában. Ezek sorai megfelelnek egy hagyományos kotta sorainak, a sorok elemei pedig tartalmazzák az egymást követő hangok magasságát és időtartamát.

```
T1/2 K2b [8G2 8G2 8G2 8G2] |K2b [8H2 8H2 8A2 8G2] |K2b 8A2 4F2. |
     K2b [8G2 8G2 8A2 8G2] |K2b [8F2 8E2 8D2 8C2] |K2b 8D2 4H1. |
     K2b 4D2 [8D2 8C2] |K2b [8H1 8H1 8C2 8H1] |K2b 8A1 4G1. |
     K2b [8H1 8H1 8A1 8G1] |K2b [#8F1 8G1 8A1 8H1] |K2b 4G1 R4 |
```

4.1. Fájl. *Egy kódolt kotta*

A 4.1-es fájlban egy ilyen speciálisan kódolt kottát láthatunk. A fájl első eleme az ütemek hosszát határozza meg. Az ütemek között (az igazi kottához hasonlóan) függőleges vonal (|) található, a kottában is szereplő gerendákat pedig nyitó és záró szögletes zárójel ([és]) jelöli, de ezek számunkra nem hordoznak lényeges információt.

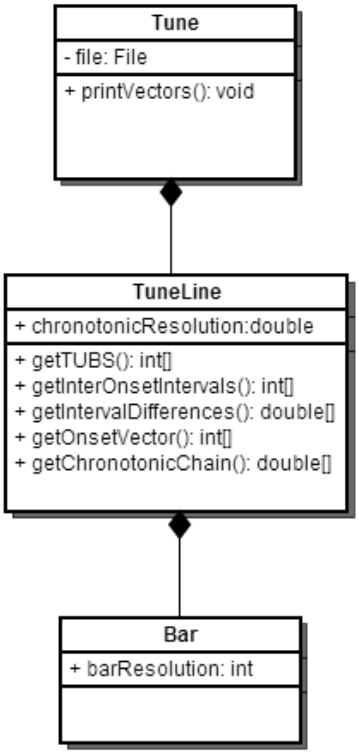
Az ütemek első eleme opcionális hangmagasság-módosító, ezért a ritmus vizsgálatakor figyelmen kívül hagyhatjuk. Utána jönnek a hangok, melynek első része határozza meg a hang hosszát, második a magasságát. A 4.1-es fájl első hangot reprezentáló eleme a $8G2$, mely egy 1/8 hosszúságú G hangot jelöl a második oktáv magasságában.

A félhangot módosító, tetszőleges számú b, vagy $\#$ közvetlenül a hang előtt is állhat, ahogy az látható a negyedik sor második ütemében szereplő $\#8F1$ hangnál. Az R-rel kezdődő elemek szüneteket jelölnek, melyek hosszát a mellettük álló szám jelöli (pl. az utolsó sor végén látható $R4$ egy 1/4 hosszú szünetet takar).

Pontozott hangok itt is előfordulhatnak, a pont helye a hangot jelölő kód utolsó pozíciója, szerepe pedig a hang hosszának másfélszerezése (ilyen pl. az első sor utolsó eleme: $4F2.$, melynek hossza így $1/4 + 1/8 = 3/8$).

4.1.1. A dalokat reprezentáló objektumok

Az egyes fájlokban tárolt adatokkal megadott dalokat egy objektum-hierarchiába képeztem le (4.1 ábra).



4.1. ábra. *A dalokat reprezentáló objektumok hierarchiája*

A teljes dalhoz tartozó osztály a `Tune` nevű osztály, mely tartalmazza a hozzá tartozó fájl referenciáját is. A `Tune` osztály `TuneLine` objektumokból épül fel, melyek mindegyike egy-egy sort reprezentál az adott fájlban. A sorok ütemekből épülnek

fel, ugyanígy egy `TuneLine` objektum az ütemet reprezentáló `Bar` típusú objektumok kompozíciója.

Minden szinten találhatóak olyan mezők, melyek az adott szinten értelmezhető ritmusvektor kiszámításához szükségesek. Az alkalmazásban a `TuneLine` osztály lett a felelős a vektorok `Bar` objektumból származó részleteinek összeállításáért.

Mivel nagyon gyakori, hogy a dalok ritmusa nem, vagy alig változik a sorok között, az a döntés született, hogy az első sort választjuk ki az összehasonlítások alapjául, majd ha szükséges, további elemzéseket futtatunk a többi sorra.

4.2. A forrásfájlok katalógusa

Az adatbázis felépítéséhez szükséges fájlok egy hierarchikus könyvtárstruktúrában találhatóak. A témával foglalkozó korábbi eljárások [8] egy katalógusfájl segítségével azonosítják a dalokat, mely egy egyszerű kulcs-érték párosnak fogható fel. A kulcs a fájlban elfoglalt sor sorszáma (1-től indulva), az érték pedig a sor tartalma, mely egy relatív útvonalat tartalmaz a fájlhoz (4.2-es fájl).

```
data/kotta/par1.knf
data/kotta/ahol2.knf
data/kotta/felj3.knf
data/kotta/megr4.knf
data/kotta/enm5a.knf
data/kotta/enm5b.knf
data/kotta/enm5c.knf
```

4.2. Fájl. *A katalógusfájl részlete*

Az itt látható $data/kotta/par1.knf$ bejegyzés sorszáma például 1, és így tovább a többi értelemszerűen.

4.3. Ritmusábrázolás

A fájlokból első lépésben a korábban ismertetett TUBS ábrázolású vektorokat nyerjük ki. Ehhez meghatározunk egy n egész számot, mely lehetőleg 2 hatvány, majd azt mondjuk, hogy az egységnyi ütemet ennyi üres dobozra osztjuk. Ezután beletesszük az elemeknek megfelelő 1-eseket a pozíciójukba. 16-os felbontást feltételezve a 4.1-es kotta első sora így ábrázolható:

$$\left(1\ 0\ 1\ 0\ 1\ 0\ 1\ 0 \mid 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0 \mid 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0 \right)$$

hiszen a nyolcadhangok ekkor kettő helyet foglalnak el a listában, az utolsó pontozott negyed pedig hatot.

Az ugyanehhez a sorhoz tartozó intervallumok hosszait reprezentáló vektor a pedig a következőképp írható fel:

$$\left(\begin{array}{cccc|cccc|cc} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 6 \end{array} \right)$$

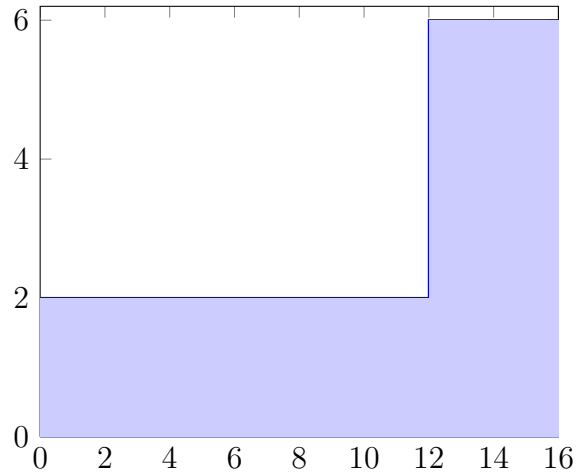
amely az előző vektorból könnyen számolható az 1-esek közötti 0-k számának megszámolásával (+1).

A szomszédos elemekhez tartozó arányok az intervallumok hosszait reprezentáló vektorból számíthatók a szomszédos elemek hányadosának sorba vételével:

$$\left(\begin{array}{cccc|cccc|cc} 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 3.00 & 0.33 \end{array} \right)$$

A hangsúlyos elemek pozícióit tartalmazó vektor:

$$\left(\begin{array}{cccc|cccc|cc} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \end{array} \right)$$



4.2. ábra. *A példa fájl kronotonikus lánca*

A kronotonikus lánc pedig:

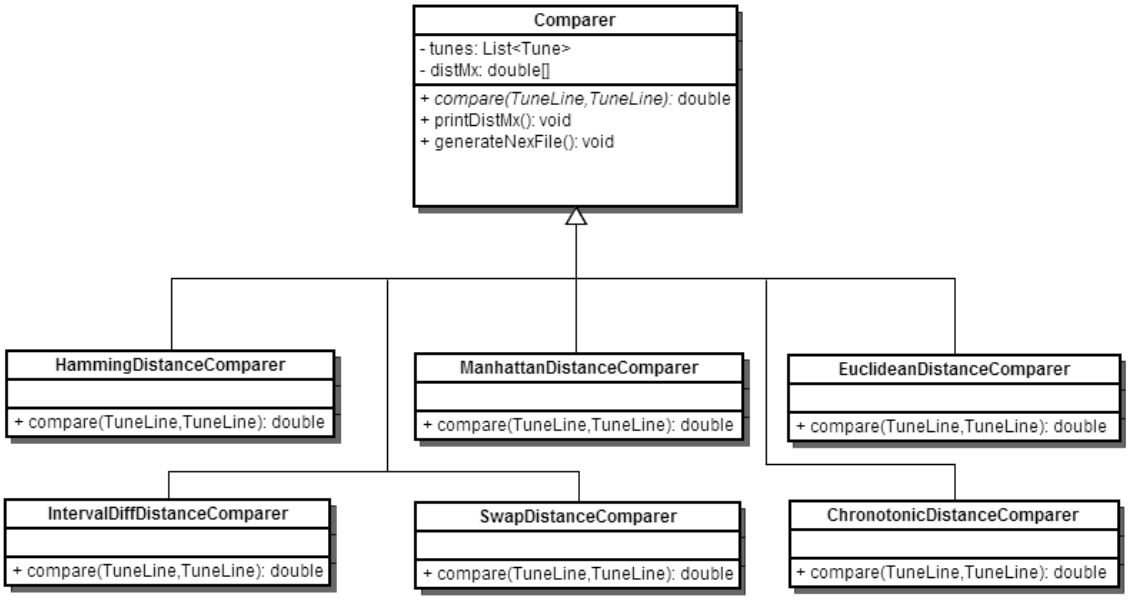
$$\left(\begin{array}{cccccccc|cccccccc|cccccccc} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 6 & 6 & 6 & 6 & 6 & 6 \end{array} \right)$$

melyet a 4.2 ábrán láthatunk grafikusan ábrázolva.

4.4. A komparátorok

A ritmusok reprezentációinak elkészítése után a következő lépés az összehasonlításért felelős osztályok megírása volt. Az objektum-orientáltság szellemében ezek egy közös, `Comparer` nevű absztrakt osztály gyermekei (4.3 ábra), mely az összehasonlító

metóduson kívül minden más segédeljárást tartalmaz (ilyen pl. a számított távolságmátrix formázott kiírása, fájlba mentése, stb.).



4.3. ábra. *A komparátor objektumok hierarchiája*

A következő osztályok implementációi felelnek meg egy-egy korábban definiált ritmikus összehasonlító módszernek:

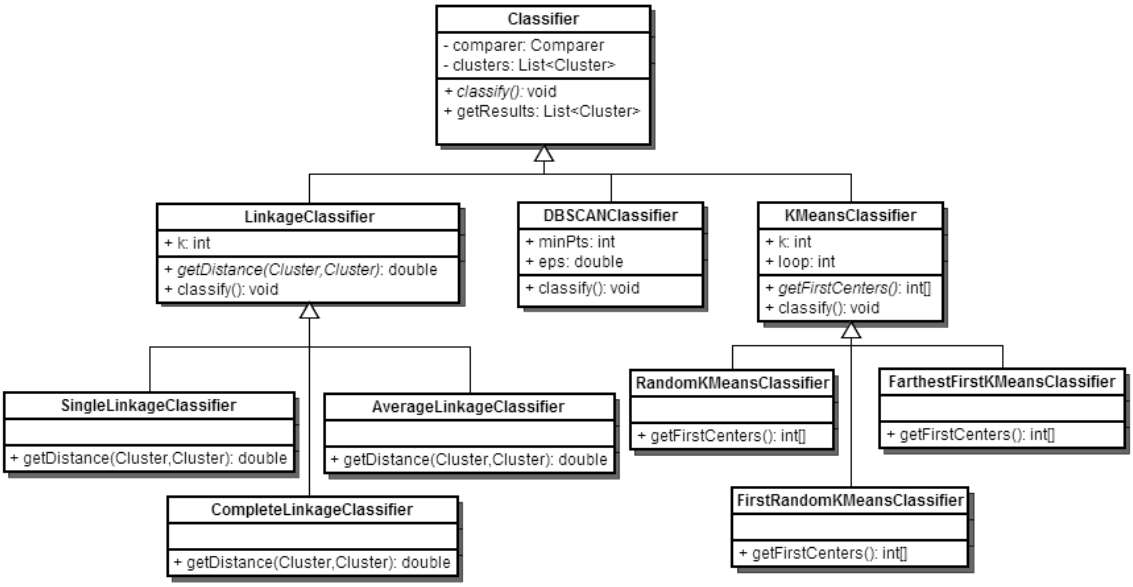
- `HammingDistanceComparer`
- `ManhattanDistanceComparer`
- `EuclideanDistanceComparer`
- `IntervalDiffDistanceComparer`
- `SwapDistanceComparer`
- `ChronotonicDistanceComparer`
- `ContinousChronotonicDistanceComparer`

4.5. A klaszterezések

A klaszterező algoritmusok mindegyikének szintén egy-egy osztály felel meg. Ezek az osztályok a `Classifier` közös absztrakt ősosztályból származnak, név szerint:

- `SingleLinkageClassifier`
- `CompleteLinkageClassifier`

- `AverageLinkageClassifier`
- `RandomKMeansClassifier`
- `FirstRandomKMeansClassifier`
- `FarthestFirstKMeansClassifier`
- `DBSCANClassifier`



4.4. ábra. *A klaszterező objektumok hierarchiája*

A rendelkezésre álló hierarchikus klaszterezés alapvető implementációi (single, complete és average linkage) egy köztes ősből (`LinkageClassifier`) származnak, és csak abban az egy metódusban térnek el, hogy hogyan határozzák meg a következő összevonandó klasztereket (4.4 ábra).

Ugyanez mondható el a különböző particionáló módszerekről is, ahol a kezdeti klaszterezés megválasztásától eltekintve minden funkcionalitást a közös ős, a `KMeansClassifier` osztály tartalmaz.

4.5.1. A klaszterezések jóságának mérése

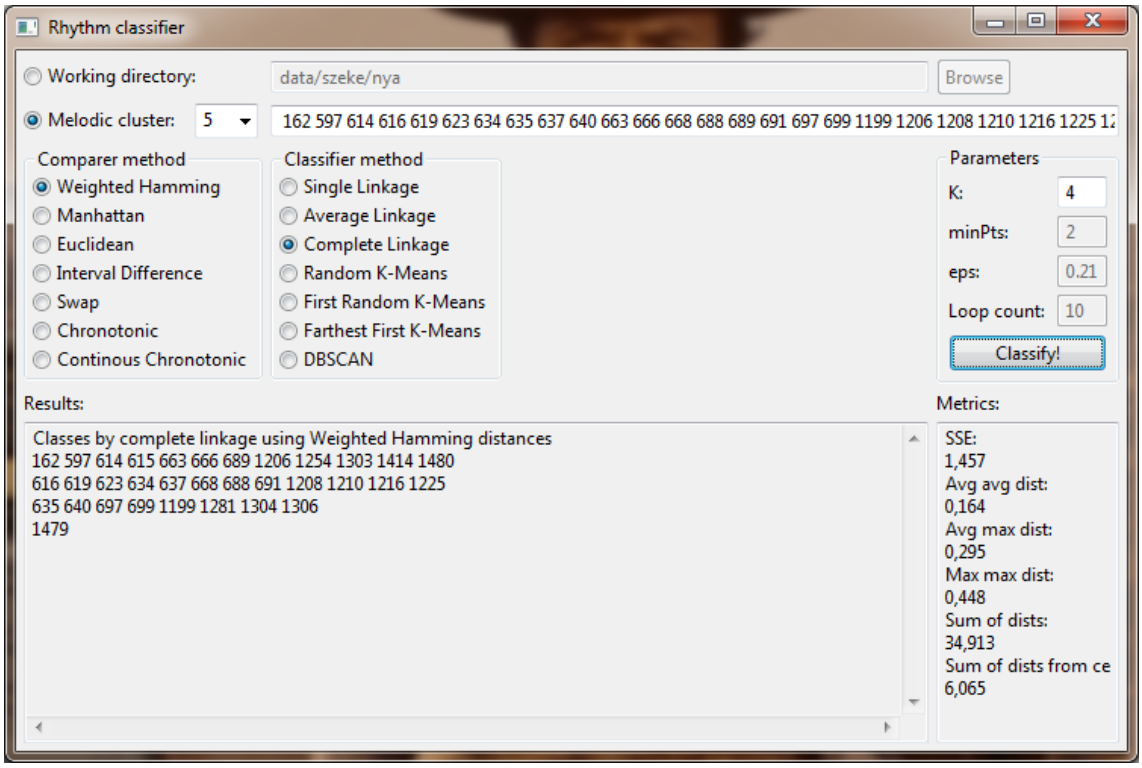
Mind a k-means, mind a DBSCAN implementációi igényelnek egy célfüggvényt, melyet minimalizálni szeretnének. Számos, 1.3.1 szakaszban definiált metrikát megvalósítottam, melyek a `MetricsUtils` osztály statikus metódusai közé kerültek. Mindegyikük egy paraméterként kapott klaszterlista alapján egy dupla pontosságú lebegőpontos nemnegatív értéket ad vissza.

A választott nyelv által megkövetelt objektumorientáltság alapelve, hogy az adatok és a hozzájuk tartozó manipuláló eljárások egy objektumba legyenek foglalva. Ezt szem előtt tartva bizonyos, adott klaszterre jellemző mértékek (pl. klaszterátmérő, klaszteren belüli távolságok összege) kiszámításáért a `Cluster` nevű segédosztály lett a felelős, mely egyébként a klaszterpéldányokat reprezentálja (mezői: középpont, elemek, stb. . .).

4.6. Futtató felület

Az alkalmazás számos futtatási módjának és azok paramétereinek könnyű kezelhetősége miatt grafikus felületet kapott, melyet az Eclipse SWT (Standard Widget Toolkit) grafikus könyvtárát használva valósítottam meg.

Az SWT az operációs rendszer grafikus megjelenítést végző szubrutinjai fölötti natív objektumok Java csomagolóinak halmazának tekinthető [23]. Minden platformon elérhető és mindegyik különbözőképp néz ki, illeszkedve az adott környezetre jellemző kinézetre. A 4.5-ös ábrán látható a fejlesztéshez használt Windows 7-es operációs rendszeren kapott felület.

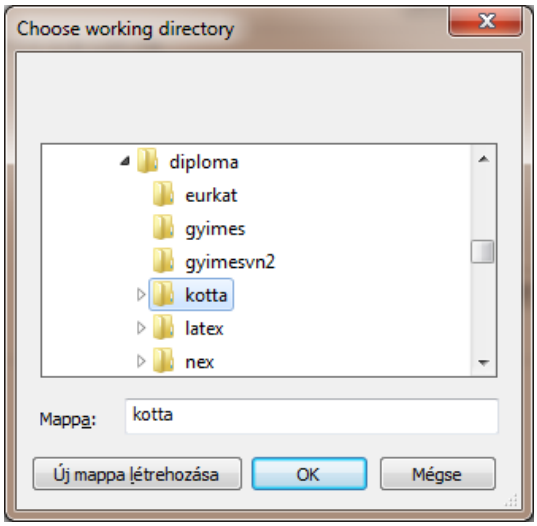


4.5. ábra. *A grafikus futtató felület*

Az ablak tetején található eszközök segítségével választható ki a forrásadatbázis helye, melyet egy könyvtárválasztó párbeszédablak (4.6 ábra) tesz könnyűvé. Az adatbázis a megadott könyvtárban és annak összes alkönyvtárában található kódolt

kották alapján épül fel, így a könyvtárstruktúrában hierarchikusan elrendezett fájlok külön csoportonként is vizsgálhatóak.

Lehetőség van arra, hogy a fájlokat a katalógusfájlban szereplő azonosítójuk segítségével egyesével adjuk meg, így a fájlhierarchiát nem követő csoportokat is elemezhetünk. A korábbi kutatásokból származó dallam-alapú klaszterbe tartozó dalok azonosítóinak megadását legördülő menü teszi még egyszerűbbé.



4.6. ábra. *Mappaválasztó párbeszédablak*

Az ablak közepén találhatóak a különböző ritmus-összehasonlítási, illetve a klaszterezési módszerek közötti választást lehetővé panelek. Ezek pontosan egy összehasonlító módszer és pontosan egy klaszterező algoritmus kiválasztását engedik meg.

A kiválasztott klaszterezés paraméterei a jobb oldali paraméterszerkesztő panelen állíthatóak. A panelen minden lehetséges paraméter szerepel, de csak azt lehet állítani, amely a kiválasztott klaszterezés számára szükséges. A hierarchikus (single, complete, average linkage) módszerek esetében ez csak az osztályok számát (K) jelenti, a k-means különböző típusaihoz ehhez még a *Loop count*-ot is be kell állítanunk (hány próbálkozásból válassza ki a minimális négyzetes hibaösszegűt), míg a DBSCAN kiválasztása csak a $minPts$ és az ε megadását igényli.

A futást a *Classify!* feliratú gomb megnyomásával indíthatjuk, az eredmények megjelenítése pedig az alsó, csak olvasható kimeneti panelen történik. A kapott klaszterek külön sorba kerülnek, az elemeiket pedig a katalógusfájlból származó azonosítók jelölik.

Az ablak jobb oldalának alsó részén helyezkedik el az eredményül kapott klaszterezésre számított metrikákat tartalmazó panel. Itt nyomon követhetjük a négyzetes távolságösszeg, maximális átmérő, stb. . . változását a különböző paraméterekre.

4.7. Távolságmátrixok szerializálása és betöltése

Abban az esetben, ha nem kívánjuk gyakran változtatni a vizsgált adathalmazt, érdemes lehet az elemek közötti távolságokat csak egyszer meghatározni, majd fájlba írni. Ezzel megspóroljuk, hogy az esetleges későbbi, különböző paraméterű klaszterezések újra és újra előállítsák ugyanazt a távolságmátrixot.

Ehhez a funkcióhoz mind a távolságmátrix szerializálását, mind a fájlból való visszatöltését implementálni kellett.

4.8. Nexus fájlok generálása

Az elkészült szoftver rendelkezik azzal a funkcióval, hogy speciális, úgynevezett Nexus fájlokat generáljon, amelyek a következő fejezetben ismertetett SplitsTree nevű alkalmazásnak adhatóak át bemeneti fájlként. Ezekben a NEX kiterjesztésű fájlokban is megtalálható a távolságmátrix, illetve megadott szabályok szerint előállított metaadatok.

Egy ilyen fájlra láthatunk példát az F.1. számú függelékben.

4.9. Felhasznált technológiák

A programozási folyamat során gyakran találkoztam olyan problémával, melyre annak általánossága miatt már jól ismert és publikusan hozzáférhető megoldások érhetőek el. Az alábbi lista tartalmazza a megvalósítás és a dokumentálás során felhasznált technológiákat.

- A katalógusfájl beolvasása során szükségem volt egy kétirányú `Map` implementációra, melyre az Apache Commons Collections könyvtárának `BidiMap` osztálya nyújtott megoldást.
- A debug során sok segítséget nyújtott a prioritási szintenkénti naplózást lehetővé tevő Log4j csomag.
- A katalógusfájl soronkénti feldolgozását és a fájlba író műveleteket a Google Guava könyvtárának fájlműveleteket egyszerűsítő eljárásaink segítségével oldottam meg.
- A grafikus felület létrehozásához az Eclipse SWT grafikus könyvtárát használtam.
- Az Apache Commons IO könyvtára a rekurzív fájlbeolvasásban segített.

- Az UML diagramok a Gliffy nevű online diagramszerkesztő alkalmazás segítségével készültek.
- Az ábrákat a Pixlr nevű online képszerkesztővel színeztem ki.
- A munka során végig verziókezelő rendszer segítségével naplóztam az elvégzett feladatokat és tartottam biztonságos helyen a forráskódot. Ehhez a `http://code.google.com` által rendelkezésre bocsátott alkalmazásokat és tárhelyet használtam.
- Az Eclipse Metrics pluginja segítségével követtem a projektet alkotó osztályok számának, méretének és komplexitásának növekedését.
- A kották MuseScore nevű kottaszerkesztő program segítségével készültek.
- A dolgozat LaTeX-ben készült.

5. fejezet

Eredmények

Az elméleti alapok és a megvalósítás részletei után lássunk néhány szép és viszonylag könnyen értelmezhető eredményt, amelyekre az elkészült alkalmazás által számított kimeneti adatokat elemezve juthatunk, illetve ellenőrizhetjük annak helyes működését. Ehhez két, egymástól eltérő vizualizációs eszközhöz fordulunk segítségért, az egyik a Multidimensional Scaling (MDS), a másik a Neighbor Net és a ráépülő SplitsTree alkalmazás.

A vizsgált csoportok alapegysége a korábbi kutatásokból származó (dallamvonal alapján számított) dallamosztály. Egy ilyen csoport nagyjából 20 és 50 közötti dalt tartalmaz.

5.1. Multidimensional Scaling

Az MDS olyan dimenziócsökkentő eljárások gyűjteménye, melyeket gyakran használnak arra, hogy vizualizációval tegyék könnyebbé adatok közti összefüggések megértését és felfedezését. A projekt korábbi szakaszában készült olyan implementáció, ami ezt alkalmazza a vizualizációra, ezért kézenfekvő, hogy kipróbáljuk az eredmények elemzése céljából.

Az algoritmus távolságmátrix alapján egy előre megadott N dimenziós térbe képzi le az elemeket. Ha ezt az N-t kellően kicsinek (2 vagy 3) választjuk, a kapott térkép emberi érzékelésünk számára is könnyen értelmezhetővé válik.

5.1.1. Az MDS működése

A két dimenzióra leképző általános algoritmus lépései a következők [24]:

1. Rendezzük el véletlenszerűen az elemeket a síkon.
2. Számoljuk ki a pontok közötti d_{ij} távolságokat.

3. A távolságmátrix elemeit egy monoton f transzformáció segítségével skálázzuk.

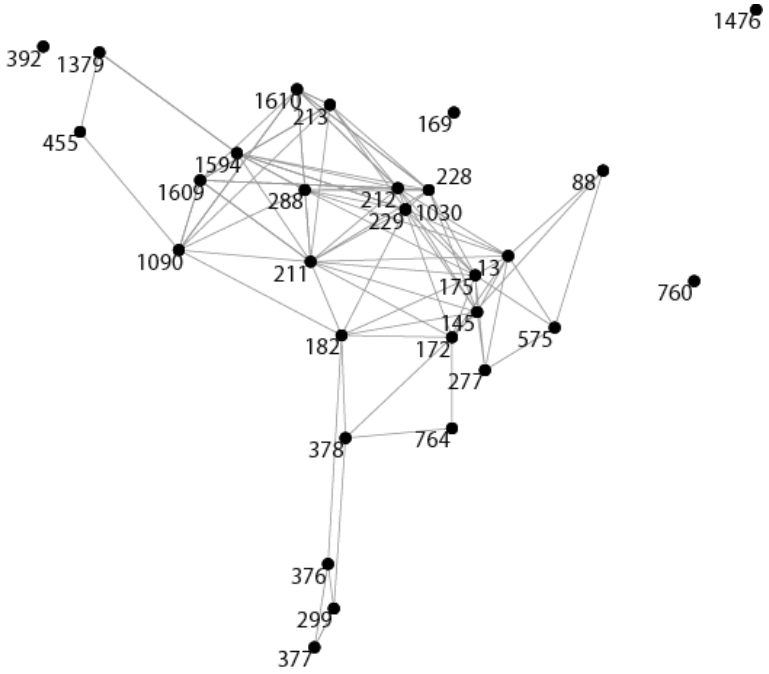
4. Minimalizáljuk az alábbi S stresszfüggvényt a pontok újrarendezésével.

$$S = \sum_{i=1}^{D} \sum_{j=1}^{D} (f(p_{ij}) - d_{ij})^2 \tag{5.1}$$

5. Ha nem elég kicsi az eredmény, folytassuk a 2-es pontnál.

Az esetünkben alkalmazott konkrét, gradiens alapú újrarendező algoritmus és az f transzformáció részletezése megtalálható a forrásjegyzékben [25].

5.1.2. Az első dallamosztály ritmikai szerkezete

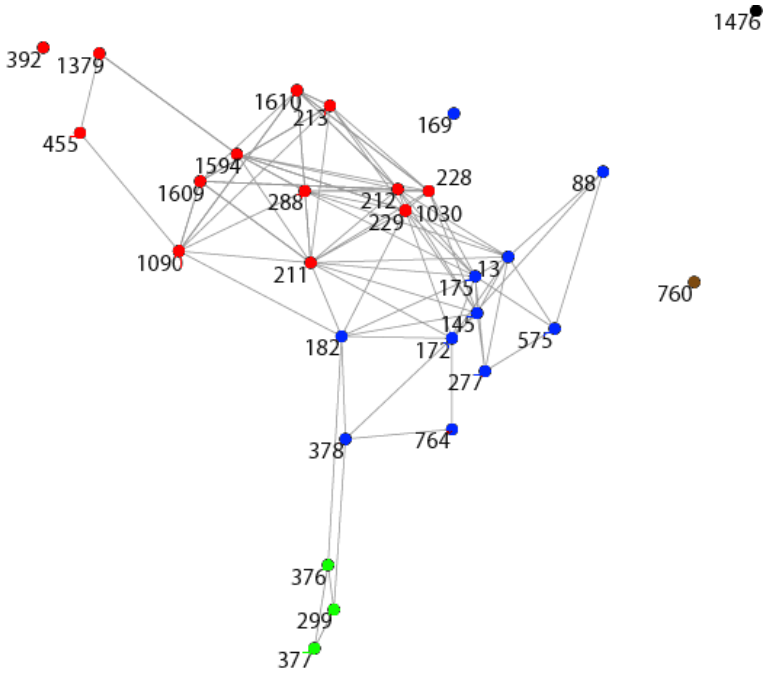


5.1. ábra. *Az 1. dallam szerinti osztály dalai MDS-sel ábrázolva (Hamming távolság alapján)*

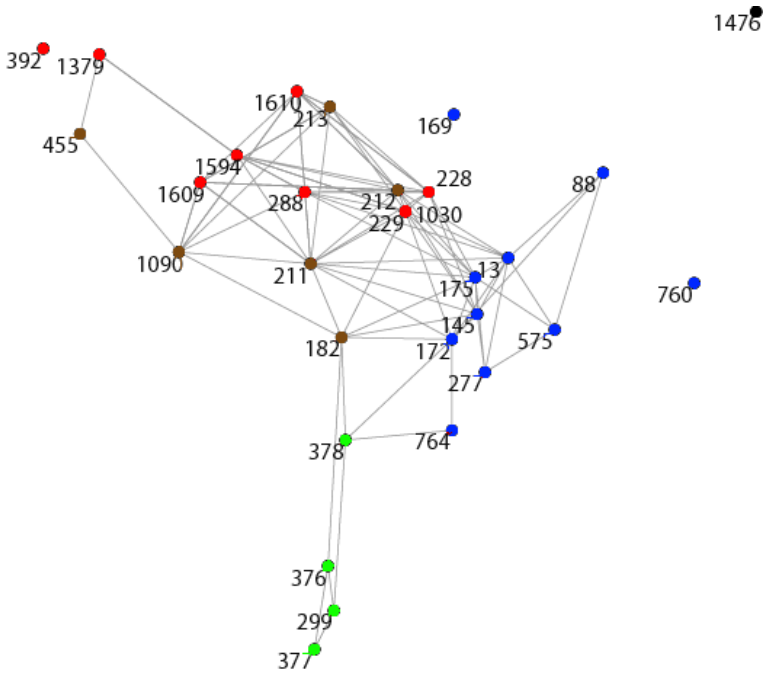
Az 5.1 ábrán látható a 30 dalból álló első dallamvonal szerinti osztály Hamming-távolság alapján számított MDS-térképe.

Meglehetősen kuszának tűnik, így nem is csoda, hogy a single linkage algoritmus 3 outlier ponton ((169), (760), (1476)) és egy kételemű csoporton (378, 764) kívül egy minden más elemet tartalmazó klasztert talál. Az average egy kicsivel jobban teljesít, az alsó fürtöt (299, 376, 377) sikeresen szeparálja a többi elemtől. A complete a nagy és sűrű klasztert felbontja két kisebb, egymást érintő klaszterre (5.2 ábra).

Az 5.3 ábrán látható, hogy a k-means a piros pontokat ránézésre véletlenszerűen még két részre bontja. Sajnos a DBSCAN algoritmus kevés értékelhető eredményt ad,



5.2. ábra. *Complete linkage szerinti klaszterek*



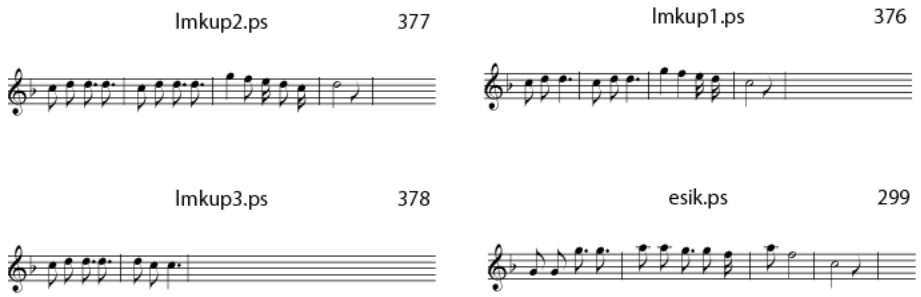
5.3. ábra. *k-means szerinti klaszterek*

ez azonban nem meglepő, hiszen a kapott térképen sem látunk igazán jól elkülönülő sűrűsödéseket.

Ha megvizsgáljuk a kapott klaszterek elemeinek kottáit, felfedezhetjük azokat a közös ritmikai tulajdonságaikat, amelyek alapján egy csoportba lettek sorolva.

Az első klaszter

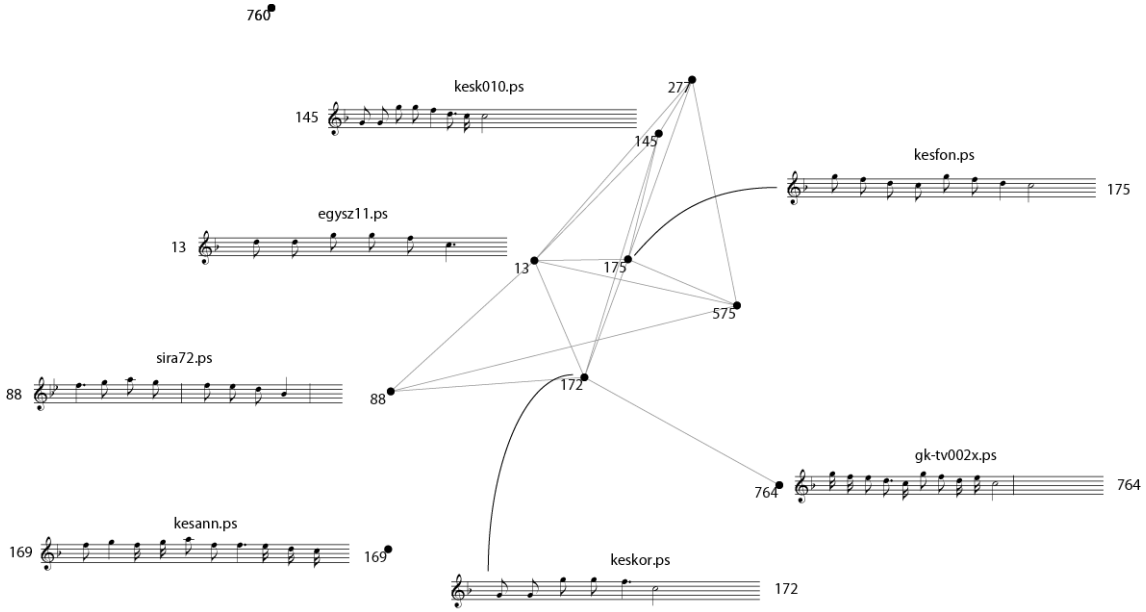
Az 5.4 ábrán látható dalok (az 5.2 ábrán zöld pontok) a gyimesi csángók sajátos páros táncának, az aszimmetrikus lüktetésű (10/16-os ütemű) lassú magyarosnak változatai [26].



5.4. ábra. *Az első klaszter kottái*

A második klaszter

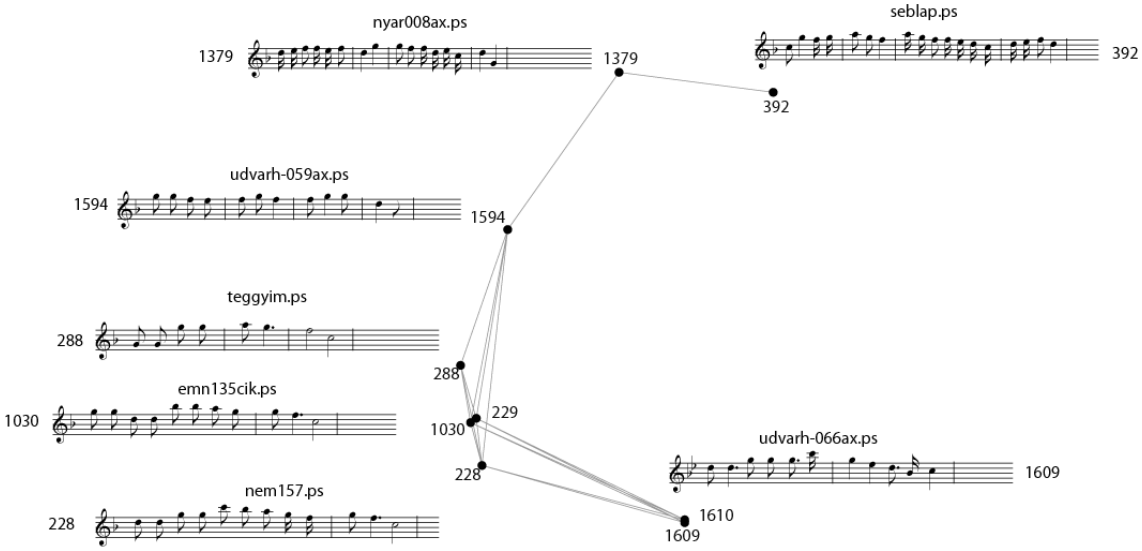
A második klaszter elemeinek (az 5.2 ábrán kékkel jelölve) kottáit láthatjuk az 5.5 ábrán. Ha szakértői szemmel vizsgáljuk őket, akkor kiderül, hogy ezek a parlando (beszédszerű előadásmódú) keservesek csoportját alkotják. Ezek a lírai dalok a lexikon [27] szerint mély lelki fájdalomból fakadó, bánatos hangvételű egyéni panaszdalok, melyeknek zenei ritmusa, a hangok időértéke teljes egészében a szövegmondáshoz igazodik.



5.5. ábra. *A második klaszter kottái*

A harmadik klaszter

A harmadik klaszterhez (az 5.2 ábra piros pontjai) tartozó kottákon ugyanannak a dallamnak a hangszeres előadásmód során variálódó ritmusát (1594-1379-392) követhetjük figyelemmel (5.6 ábra).



5.6. ábra. *A harmadik klaszter kottái*

A negyedik klaszter

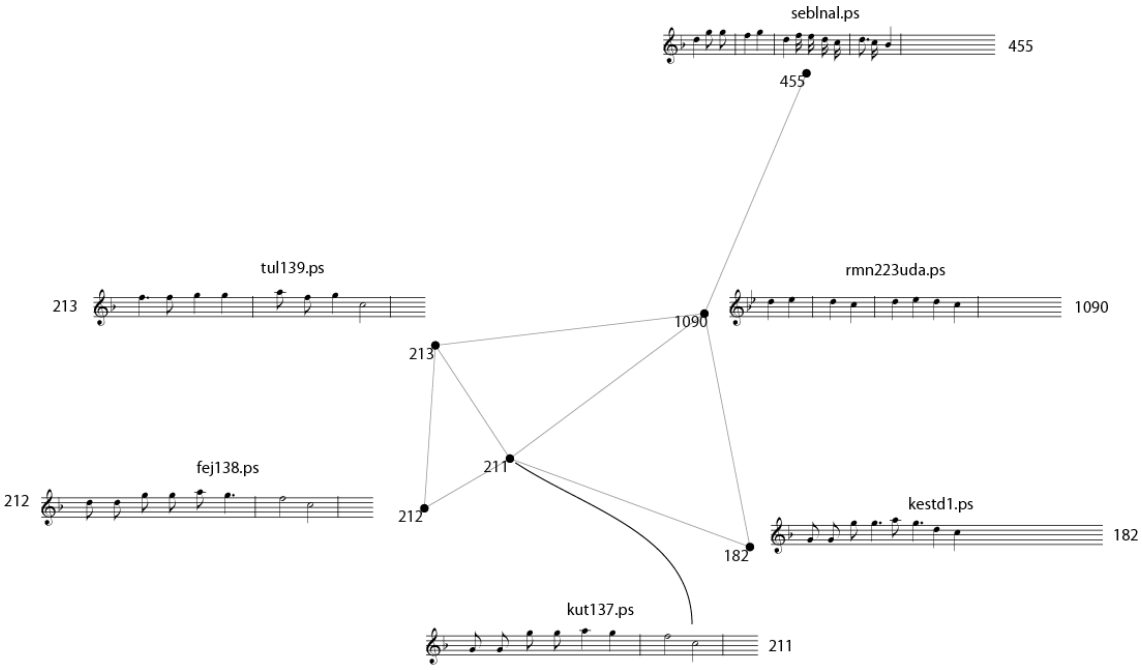
Az 5.2 ábrán barna színnel jelölt, k-means szerinti negyedik klaszter mindhárom linkage algoritmus kimenetében beleolvadt egy másik klaszterba, aminek oka a szoros kapcsolat a határaikon lévő elemek között. (Ha alaposan szemügyre vesszük, megfigyelhető, hogy a harmadik klaszterbe sorolt 288 azonosítójú dal ritmusa mennyire hasonlít a negyedik klaszterbe sorolt 212-esére).

Az F.2 számú függelék tartalmazza a második dallamosztályra kapott osztályozásokhoz tartozó kottákat azok közös tulajdonságainak rövid leírásával együtt.

5.2. A SplitsTree alkalmazása

A SplitsTree egy bioinformatikai alkalmazásokban népszerű következtető és vizualizáló szoftver, amely képes arra, hogy távolságmátrixban adott távolságokon alapuló „térképet” rajzoljon az adathalmaz elemeihez. Ehhez természetesen csak szimmetrikus távolságértékeket tartalmazó mátrixot tud értelmezni.

A szoftver speciális, NEX kiterjesztésű fájlok segítségével működik, melyek tartalmazzák a távolságmátrixot és számos más metainformációt. Ahogy a 4.8 fejezetben említettem, az elkészült alkalmazás képes ilyen fájlok generálására.



5.7. ábra. *A negyedik klaszter kottái*

5.2.1. A Neighbor Net algoritmus

A SplitsTree által generált ábrák a Neighbor Net algoritmus segítségével készülnek, melynek lépései nagy vonalakban a következőek [28]:

1. Kezdetben minden elem egy különálló pont.

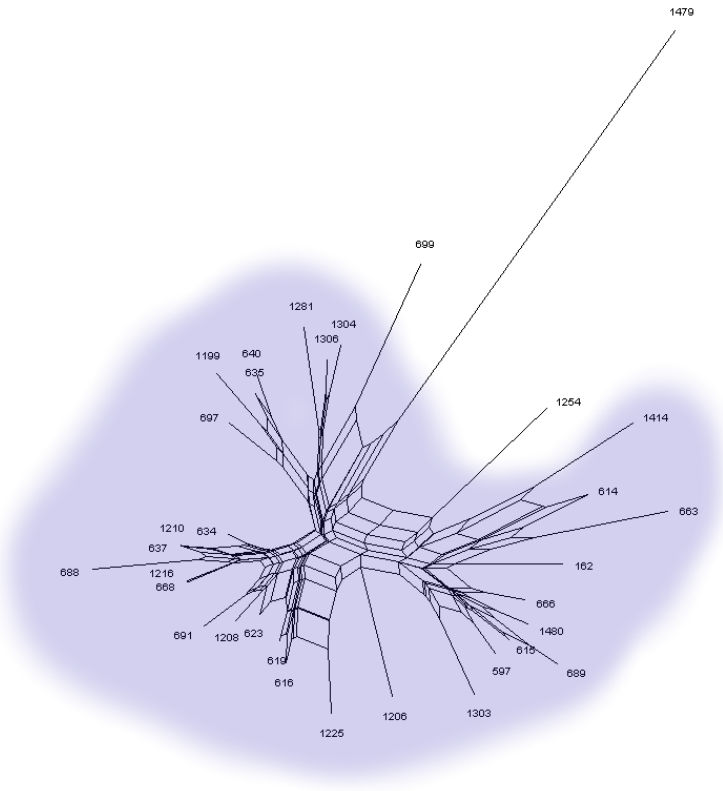
2. Minden iterációs lépésnél kiválasztunk egy pontpárt addig, amíg legalább 3 pontunk van. Ekkor azonban még nem feltétlenül vonjuk össze őket, ahogy azt tettük volna a linkage algoritmusok során. Csak akkor vonunk össze pontokat, ha másodszorra is beválasztottuk ugyanazt a pontot a párba (jelöljük ezt a pontot x-szel). Ekkor x, y és z összekötött pontokból létrehozunk két új, u és v egymással szintúgy összeköttetésben lévő pontot (melyek az ábrán belső pontok lesznek).

3. Frissítjük a távolságmátrixot: u és egy tetszőleges w között a távolság az u által reprezentált pontok (x és y) és w távolságának speciális függvénye, majd folytatjuk a kettes pontnál.

A pontpárok kiválasztásához használt formulát és a távolságmátrix redukciója során alkalmazott távolsági képletet itt nem részletezzük, megtalálható a forrásdokumentumokban [28].

5.2.2. Az ötödik dallamosztály ritmikai szerkezete

Vizsgáljuk meg, hogy a klaszterezések által adott eredmények mennyire tükröződnek a SplitsTree által rajzolt fákon! Ehhez használjuk az ötödik dallam szerinti csoport 33 elemét. Ekkor az alapértelmezett K az 1.17 képlet szerint $\sqrt{\frac{33}{2}} \approx 4$.

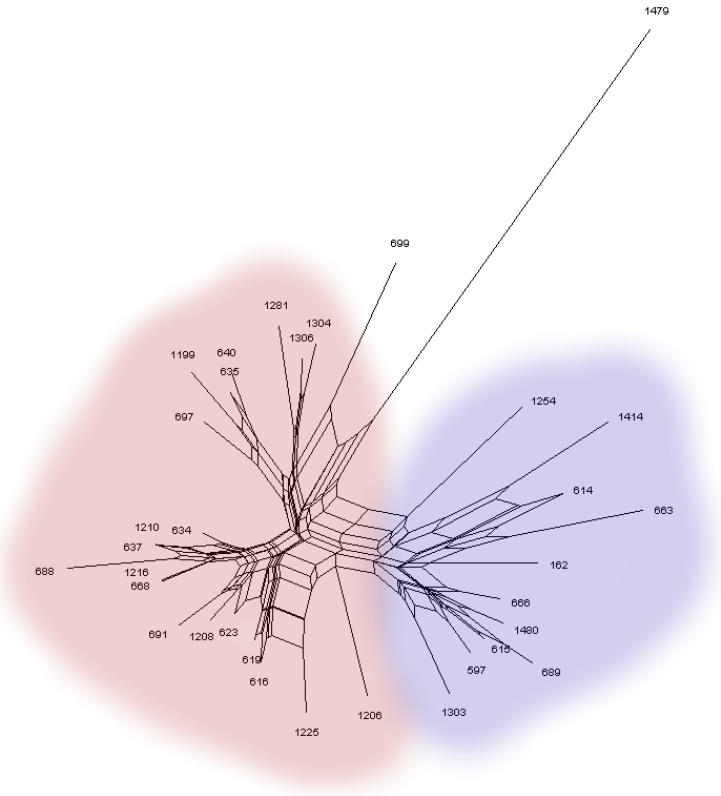


5.8. ábra. *Single linkage szerinti színezés*

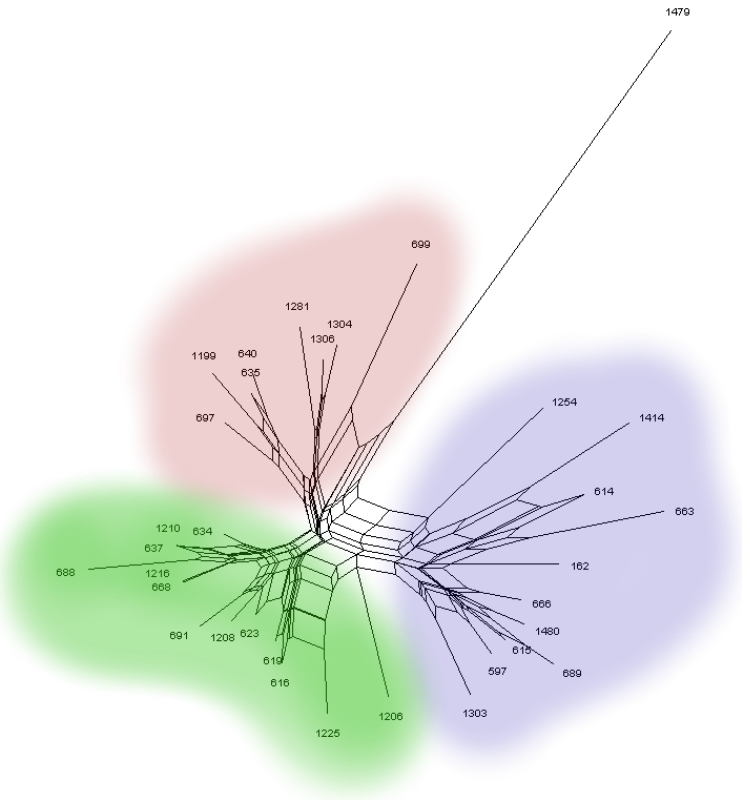
Az 5.8 ábrán látható, hogy mi történhet túl közeli elemeket tartalmazó klaszterek esetén a single linkage algoritmussal: egy nagy klaszterbe mosódnak össze az elemek (és kapunk 3 mindenkitől távolra eső outlier pontok alkotta egyelemű klasztert). Ahogy korábban említésre került, erre megoldást nyújthat egy maximális távolságérték megadása, amelyet elérve már nem húzunk össze klasztereket, de ennek a paraméternek a kiszámítása nem könnyű, ráadásul adathalmazonként változik is.

Az 5.9 ábra mutatja, hogy az average linkage algoritmus ennél hatékonyabb, hiszen használatával két egymástól jól elkülönülő, nagy klasztert és két, egy-egy outlier pontból álló klasztert kapunk.

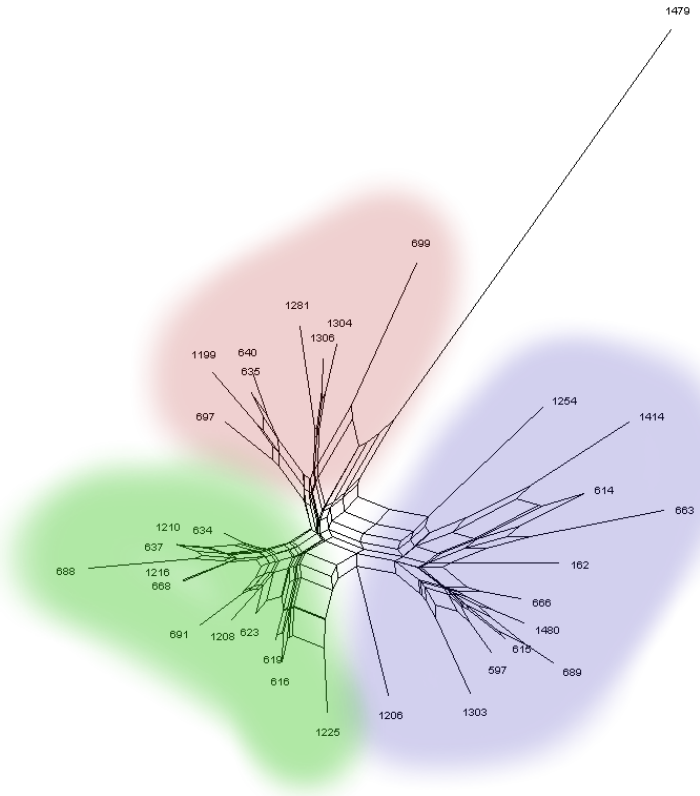
A complete linkage klaszterezés tűnik a legjobb választásnak, hiszen ez az, amelyik a szemmel látható 3 fő klasztert pontosan azonosítja (5.10 ábra). Ettől egyetlen, középen lévő elem különböző besorolásával tér el a legtávolabbi elemek alkotta klaszterekkel induló k-means algoritmus (5.11 ábra) kimenete, tehát ez is megfelelő eredményt ad.



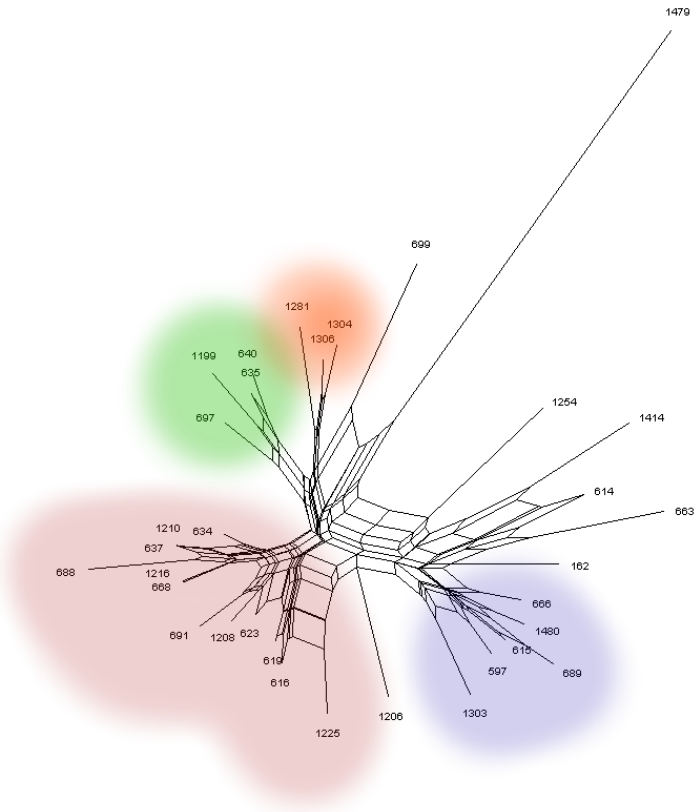
5.9. ábra. *Average linkage szerinti színezés*



5.10. ábra. *Complete linkage szerinti színezés*



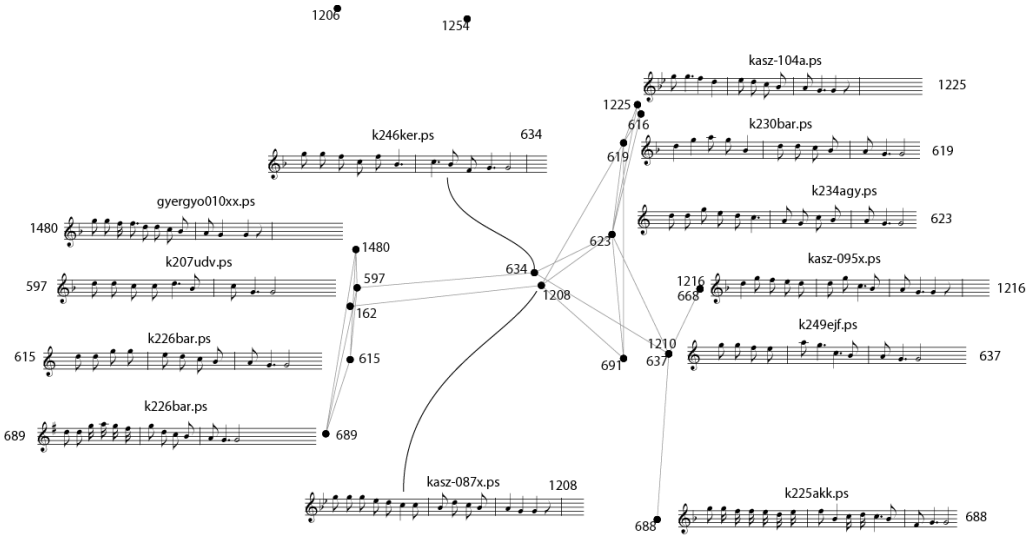
5.11. ábra. *k-means szerinti színezés*



5.12. ábra. *DBSCAN szerinti színezés*

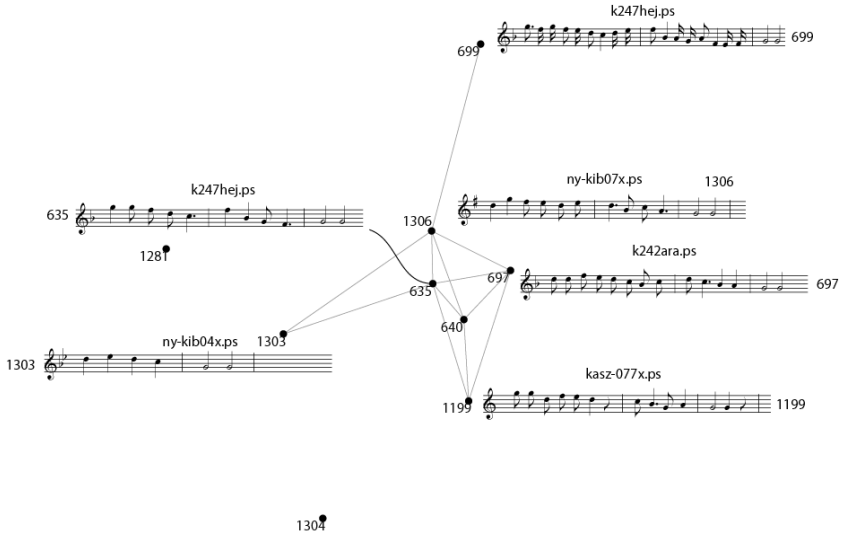
A DBSCAN algoritmus kimenete sem meglepő: eredményesen megtalálja az egymáshoz közel eső sűrűsödéseket, összesen 4-et (5.12 ábra). Több elemről állítja, hogy outlier, mint az eddigiek, de az általa adott klasztereken belüli elemek valóban nagyon közel vannak egymáshoz. Az egyetlen nehézség ezzel a módszerrel az, hogy sokat kellett „játszani" a két paraméterrel, hogy értelmezhető eredményt kapjunk (itt most $minPts = 2$ és $\varepsilon = 0.17$).

A klaszterek belső szekezete



5.13. ábra. *Az első klaszter kottái*

Ha vetünk egy pillantást a klaszterek eleminek kottáira is, kiderül, hogy az 5.13 ábrán látható dalok közös tulajdonsága a ti-táj-táá zárlat (az ezekhez tartozó dalok a fenti complete linkage szerinti 5.10 ábra kékkel és zölddel jelölt elemei).

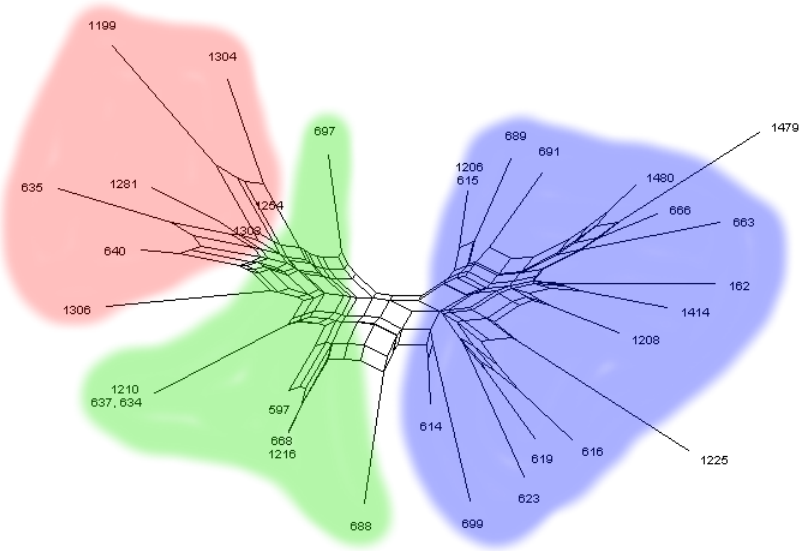


5.14. ábra. *A második klaszter kottái*

Az 5.14 ábra a táá-táá zárlatú dalok kottáit tartalmazza (ezek pedig a k-means szerinti 5.11 ábra pirossal jelölt elemei).

További távolságmetrikák használata

Tanulságos lehet megfigyelni, hogy különböző módszerrel számított távolságmátrix alapján rajzolt ábrák klaszterezései is hasonló eredményt mutatnak-e. Ehhez vizsgáljuk meg az intervallum-vektorok Euklideszi-távolsága alapján számított távolságmátrixból épült fát és rajta az average linkage szerinti színezést (5.15 ábra). A színeket úgy választottam meg, hogy látható legyen a klaszterezés eredményeinek hasonlósága.



5.15. ábra. *Euklideszi távolság alapján rajzolt fa average linkage szerinti színezése*

A Hamming-távolság és az Euklideszi-távolság szerinti klaszterek csak néhány elemben térnek el, pedig különböző vektorokra számítottuk őket. A trendek mégis mindkettőben láthatóak, a különbségek pedig annak bizonyítékai, hogy nem mindig határozható meg egyértelműen egy elem csoportja, gyakran vannak átfedések is.

További összehasonlítások alapján készült ábrákat láthatunk ugyanerre a dallamosztályra az F.3 számú függelékben.

5.3. Összegzés

Amint ezeken a példákon követhető volt, az algoritmusok meglehetősen ügyesen csoportosítják az elemeket, de ehhez megfelelő paraméterválasztás szükséges. Nem mindig bizonyul helyesnek a K-ra adott becslés, illetve a DBSCAN paramétereinek megválasztása sem triviális feladat.

6. fejezet

Összefoglalás és kitekintés

6.1. Összefoglalás

A klasszikus zenetudományi osztályozás a dallamvonal alapján történik, ám a közös dallamvonal-osztályba tartozó dallamok ritmikailag még nagyon különbözőek lehetnek. A ritmikai kapcsolatokat sorba rendezéssel a példatárak nem tudják érzékeltetni, mivel a kapcsolatrendszer több dimenziós.

A dolgozatomban bemutattam több olyan módszert, amelyek ezeket a kapcsolatokat vizualizálni tudják. A munka három fő pontban foglalható össze:

1. Áttekintettem és kipróbáltam a szakirodalomban fellehető ritmikai hasonlósági metrikákat. Ezekből elsősorban a Hamming-távolság konkrét eredményeit értékeltük zeneileg.

2. Megvizsgáltam a szóba jöhető klaszterezési eljárásokat. Ezekből elsősorban a linkage algoritmusok és a k-means eredményeit értelmeztük zeneileg. A k-means alkalmasabbnak bizonyult zeneileg értelmezhető ritmikai fürtök azonosítására.

3. Az eredmények vizualizálására az MDS és a Neighbor Net algoritmusokat próbáltam ki. Ezek közül a kottákkal kiegészített MDS térképeken mutattam be, hogy a közös dallamvonal-osztályhoz tartozó dallamok ritmikai rokonsági rendszere ezen a módon jól kimutatható. Mindkét vizualizációs módszer elkülönít különböző ritmikai fürtöket egy-egy dallamvonal-osztályon belül (5.1-5.3, 5.8-5.12, 5.15 ábrák). Ennél még egy szinttel mélyebbre is lementem, és a közös dallamvonal osztályon belül a közös ritmikai osztályba tartozó dallamok ritmikai kapcsolatait is feltérképeztem (5.4-5.7, 5.13-5.14 ábrák).

Ezzel kidolgoztam egy olyan módszert, mely a dallamok kapcsolatainak mélyebb és pontosabb kimutatására alkalmas, mint a népdalgyűjtemények lineáris, sorba rendező módszere.

6.2. Jövőbeli lehetőségek

Az elkészült klaszterező algoritmusok egyelőre csak az erdélyi Székelyföldről származó dalokat tartalmazó adatbázison futottak le. Mindenképp tervezett ennek bővítése, hogy érdekes összefüggéseket találhassunk akár a Kárpát-medence népdalkincsét, akár még bővebb adathalmazt vizsgálva.

Az adatbázis bővítése mellett a szoftver funkcionalitását is kiterjesztenénk, akár újabb ritmikus metrikák, akár eddig nem implementált klaszterező algoritmusok hozzáadásával. Bizonyos módszerek hatékonyság szempontjából is továbbgondolást igényelnek, ugyanis többszörös méretű adatbázis esetén négyzetesen nő az idő-, és erőforrásigényük. Ezekben az esetekben többszálú, elosztott módszerek jelentős sebességnövekedést is elérhetnek.

Köszönetnyilvánítás

A munka során rengeteg segítséget kaptam Juhász Zoltántól, akitől a diplomatéma ötlete is származik. Köszönet illeti még Szeredi Pétert és Katona Gyulát, akik a tanszéki adminisztrációval és kiindulási anyagokkal támogatták a dolgozat elkészülését.

A Morgan Stanley Business Metrics csapata elviselte a hiányomat és megadta a lehetőséget, hogy annyi időt töltsek a munkával, amennyit csak szeretnék. Köszönet ezért Venczel Zsoltnak és Bodon Ferencnek.

Szeretném megköszönni a Számítástudományi és Információelméleti Tanszéknek, hogy lehetővé tette, hogy a szívemhez oly közel álló népzenéhez kapcsolódó témát válasszak diplomatémaként.

Irodalomjegyzék

[1] Bartók Béla. *A népzenéről.* Magvető, 1981.

[2] Dr. Bodon Ferenc. *Adatbányászati algoritmusok.* PhD thesis, Budapesti Műszaki és Gazdaságtudományi Egyetem, February 2010.

[3] Buza Krisztián Bodon Ferenc. Adatbányászat, 2012.

[4] Lior Rokach. A survey of clustering algorithms. *Data Mining and Knowledge Discovery Handbook*, 2010.

[5] P.J. Flynn A.K. Jain, M.N. Murty. Data clustering: A review. *ACM Computing Surveys*, 31(3), 1999.

[6] Kanti Mardia. Multivariate analysis. *Academic Press*, 1979.

[7] Zsolnai Károly. Genetikus algoritmusok. Nagyméretű adathalmazok kezelése prezentáció, 2011.

[8] Juhász Zoltán. *A zene ősnyelve.* Frík Kiadó, 2006.

[9] Göncz Zoltán. Bach testamentuma. *Muzsika*, augusztus 2008.

[10] Benkő András. *A Bolyaiak zeneelmélete.* Kriterion Könyvkiadó, 1975.

[11] Hit, zene, matematika. Interjú W. R. Wade professzorral. Természet Világa, 1986.

[12] Alan Marsden. 'What was the question?': music analysis and the computer. *Ashgate*, pages 137–147, 2009.

[13] Dobszay László. *A klasszikus periódus.* Editio Musica Budapest, 2012.

[14] Sárosi Bálint. *Zenei anyanyelünk.* Planétás Kiadó, 2003.

[15] Bartók Béla. *A magyar népdal.* Rózsavölgyi és társa, 1924.

[16] Kodály Zoltán. *A magyar népzene.* Zeneműkiadó, 1981.

[17] Szabolcsi Bence. Népvándorláskori elemek a magyar népzenében: adatok a magyar népi hagyományok keleti kapcsolataihoz, 1935.

[18] Sipos János. Bartók nyomában Anatóliában. Balassi Kiadó, 2002.

[19] Vargyas Lajos. *A magyarság népzenéje*. Planétás Kiadó, 2002.

[20] Godfried Toussaint. Computational geometric aspects of rhythm, melody and voice-leading. *Computational Geometry: Theory and Applications*, 2009.

[21] Godfried Toussaint. The geometry of musical rhythm. *Lecture Notes in Computer Science*, 3742:198–212, 2005.

[22] Godfried Toussaint. A comparison of rhythmic dissimilarity measures. *FORMA*, 21(2):129–149, June 2006.

[23] Nyitrai József. Az Eclipse kiterjesztési lehetőségeinek bemutatása, 2010.

[24] Florian Wickelmaier. An Introduction to MDS. Technical report, Sound Quality Research Unit, Aalborg University, Denmark, 2003.

[25] Juhász Zoltán. A mathematical study of note association paradigms in different folk music cultures. *Journal of Mathematics and Music: Mathematical and Computational Approaches to Music Theory, Analysis, Composition and Performance*, 6(3):169–185, 2012.

[26] Szenik Ilona. Népzenetudományi jegyzetek, 1985.

[27] Magyar Néprajzi Lexikon. Akadémiai Kiadó, 1977-1982.

[28] Vincent Moulton David Bryant. Neighbor-Net: An agglomerative method for the construction of phylogenetic networks, 2003.

Ábrák jegyzéke

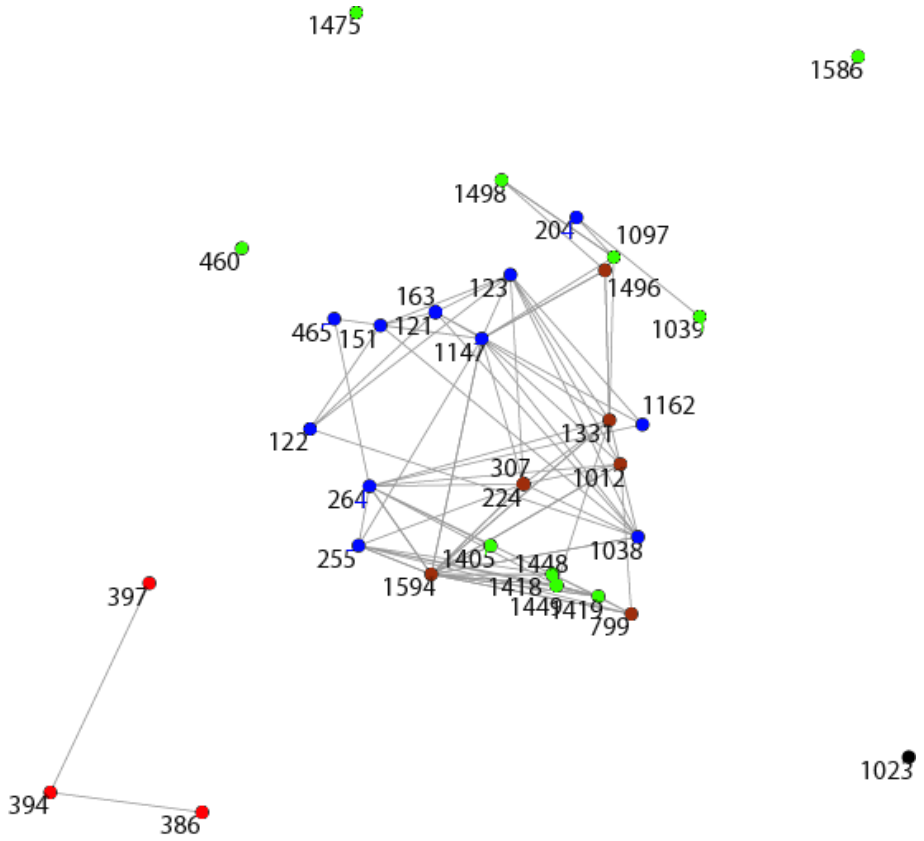
Függelék

F.1. Dallamosztályhoz generált Nexus fájl

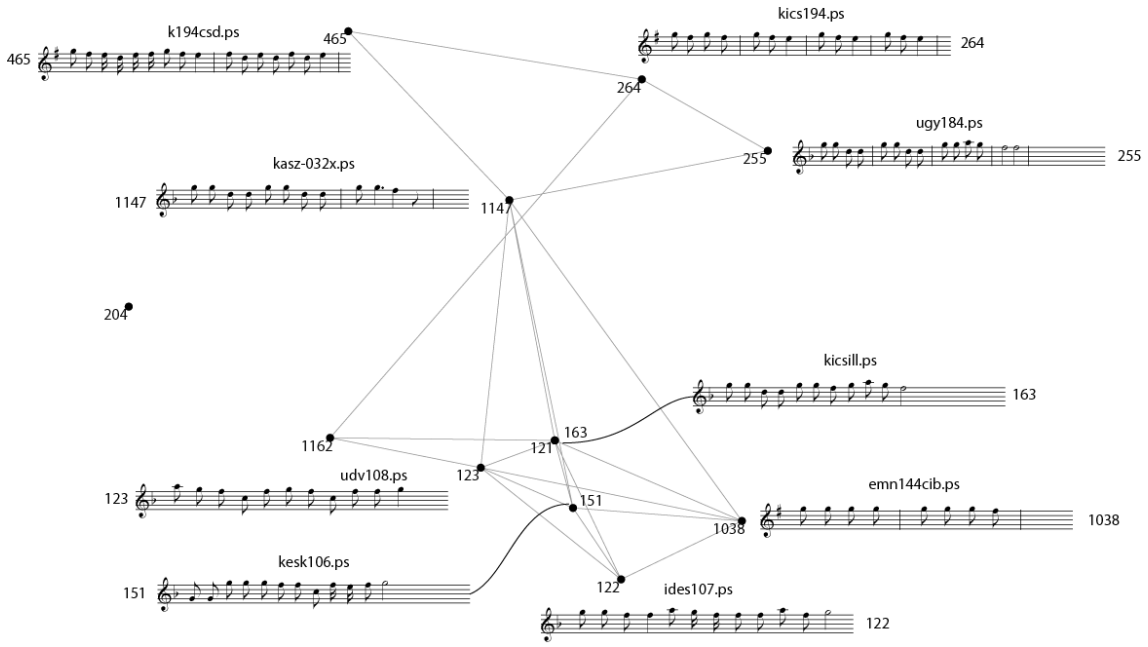
```
#NEXUS

BEGIN taxa;
        DIMENSIONS ntax=8;
TAXLABELS
        55
        315
        352
        723
        724
        1074
        1163
        1453
;
END;

BEGIN distances;
        DIMENSIONS ntax=8;
        FORMAT
                triangle=LOWER
                diagonal
                labels
                missing=?
        ;
MATRIX
        55      0.000
        315     0.125   0.000
        352     0.475   0.431   0.000
        723     0.370   0.370   0.524   0.000
        724     0.523   0.523   0.692   0.415   0.000
        1074    0.136   0.244   0.401   0.440   0.600   0.000
        1163    0.190   0.298   0.401   0.480   0.644   0.045   0.000
        1453    0.250   0.250   0.475   0.467   0.523   0.352   0.406   0.000

;
END;
```

F.1.1. Fájl. *A 21-es dallamosztályhoz generált Nexus fájl (Hamming-távolság alapján)*

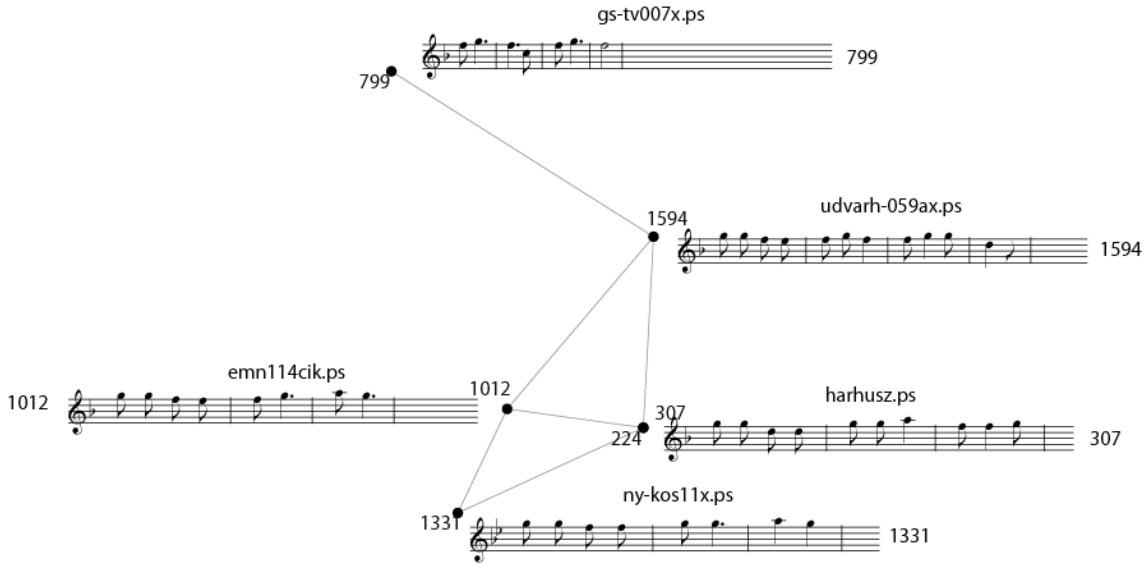
F.2. A második dallamosztály szerkezete Hamming-távolság alapján



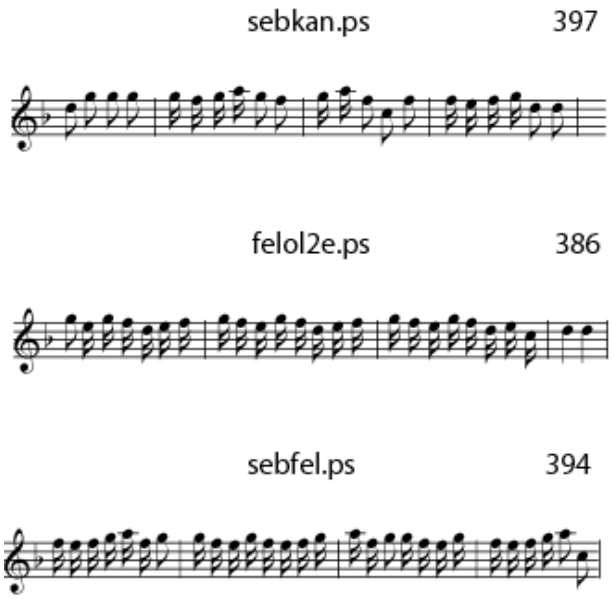
F.2.1. ábra. *A második dallamosztály MDS ábrája Hamming-távolság alapján, k-means szerint színezve*



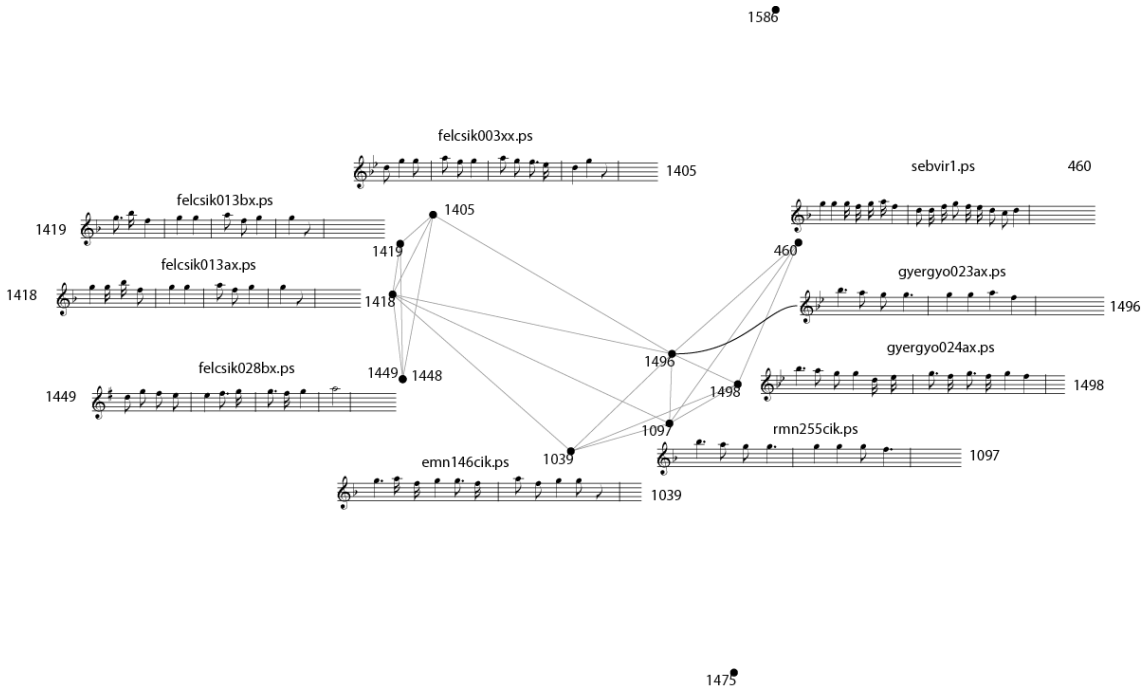
F.2.2. ábra. *Első klaszter: alul parlando keserves, felül 2*(4/4)-es és 4*(2/4)-es változatok*



F.2.3. ábra. *Második klaszter: alul 3*(2/4), fölül 4*(2/4) - a bővülés 224 és 1594 között történik*

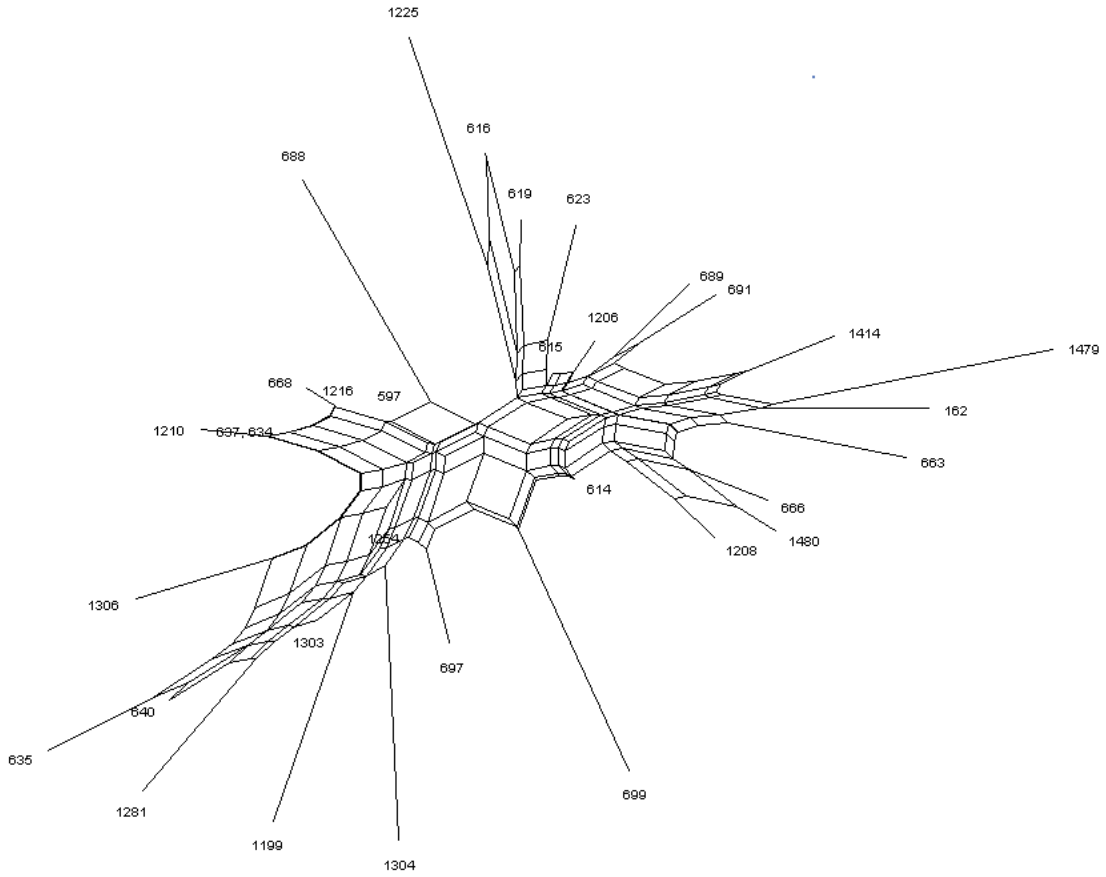


F.2.4. ábra. *Harmadik klaszter: Hangszeres kanásztánc-típus változatai*

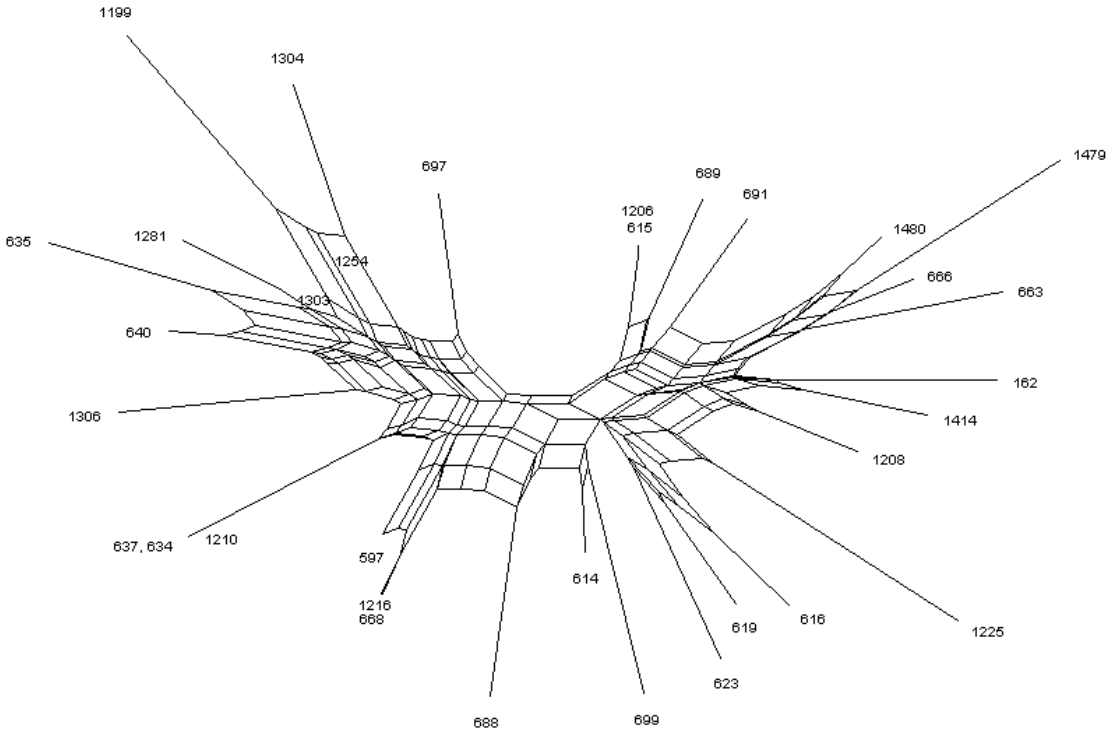


F.2.5. ábra. *Negyedik klaszter: Bal oldalon 4*(2/4), jobb oldalon 2*(4/4)*

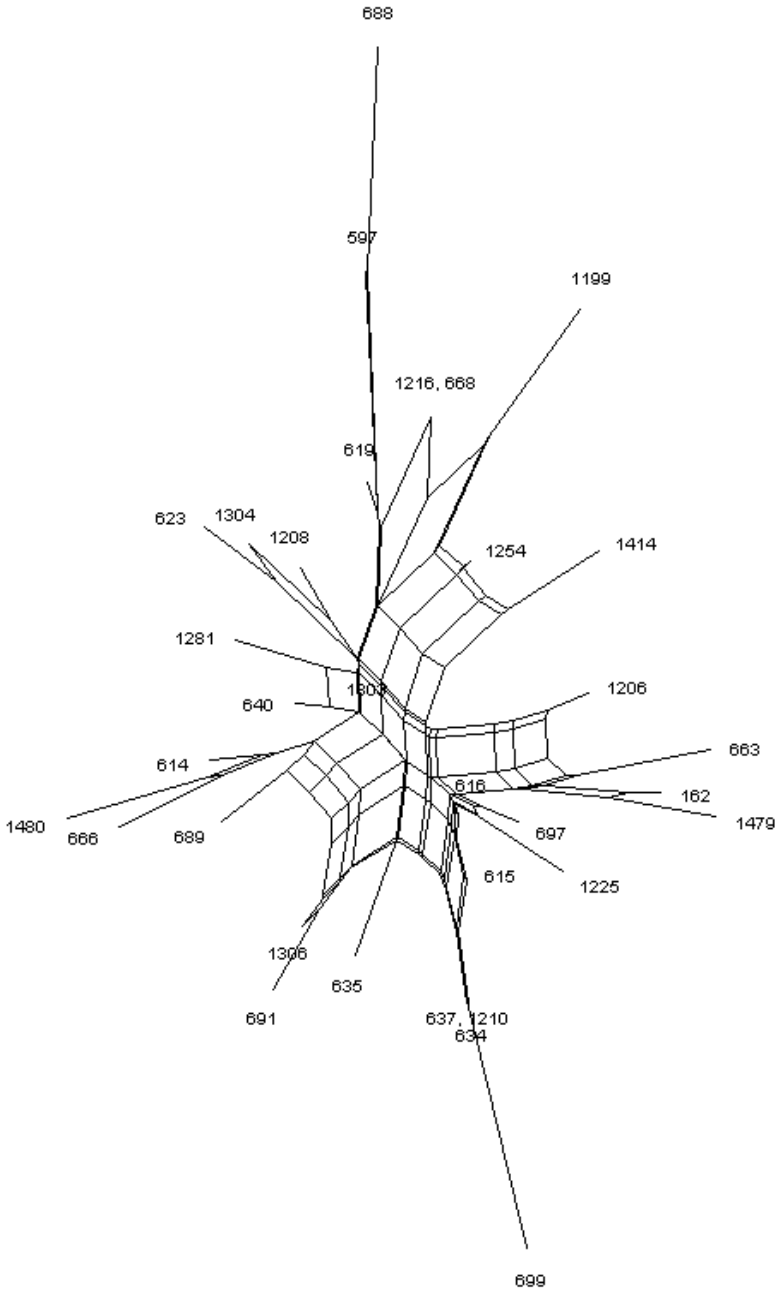
F.3. Az ötödik dallamosztály SplitsTree fái különböző távolságok alapján



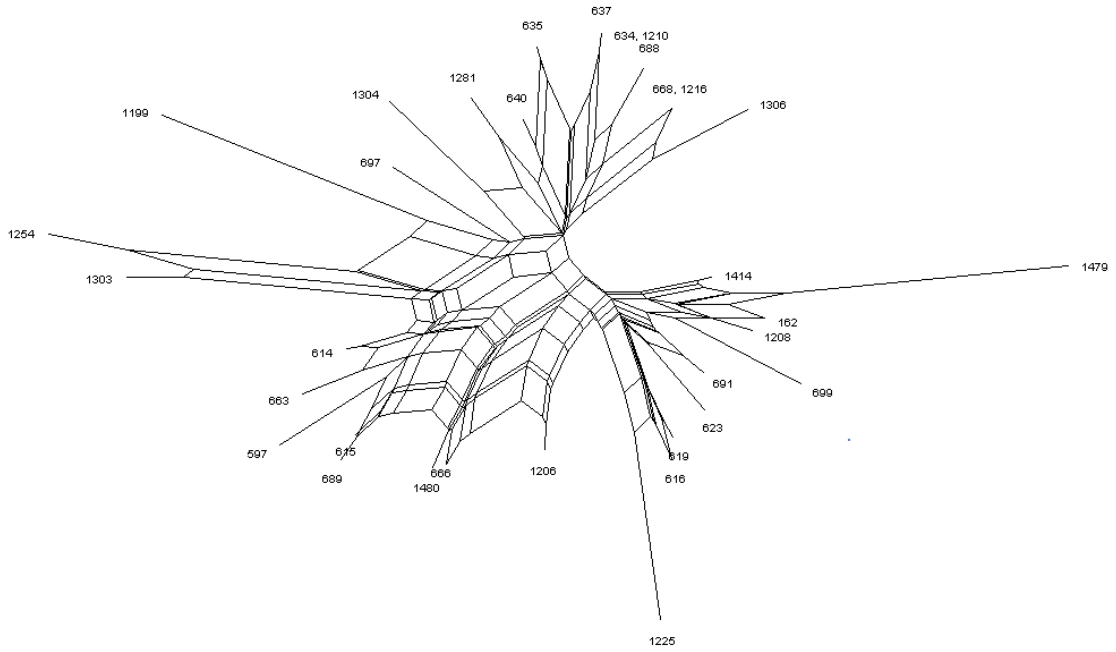
F.3.1. ábra. *Az ötödik dallamosztály elemei Manhattan-távolság alapján*



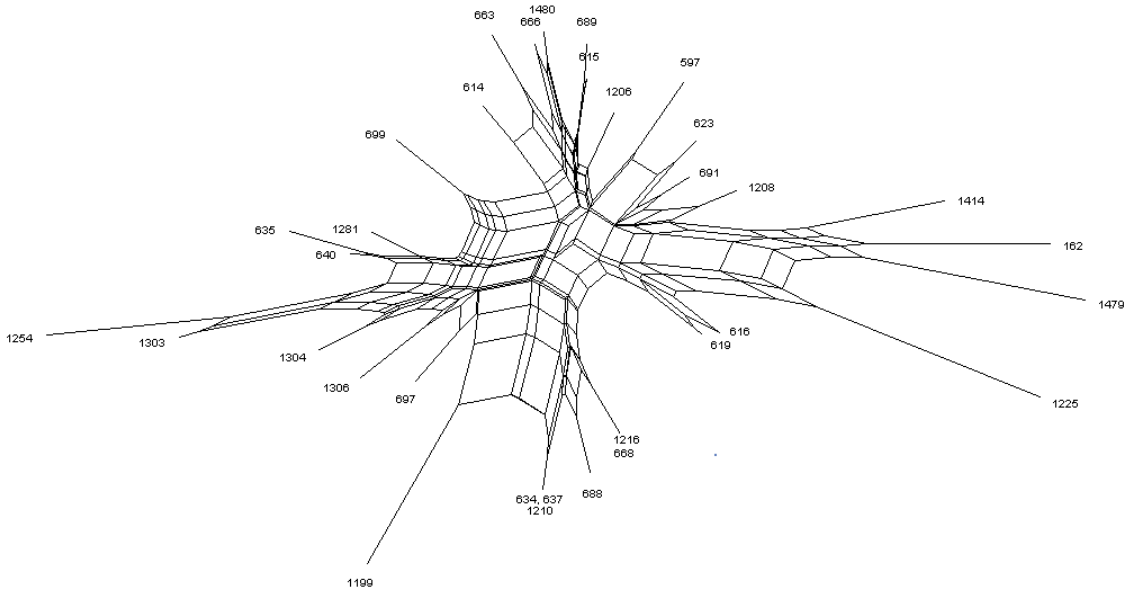
F.3.2. ábra. *Az ötödik dallamosztály elemei Euklideszi-távolság alapján*



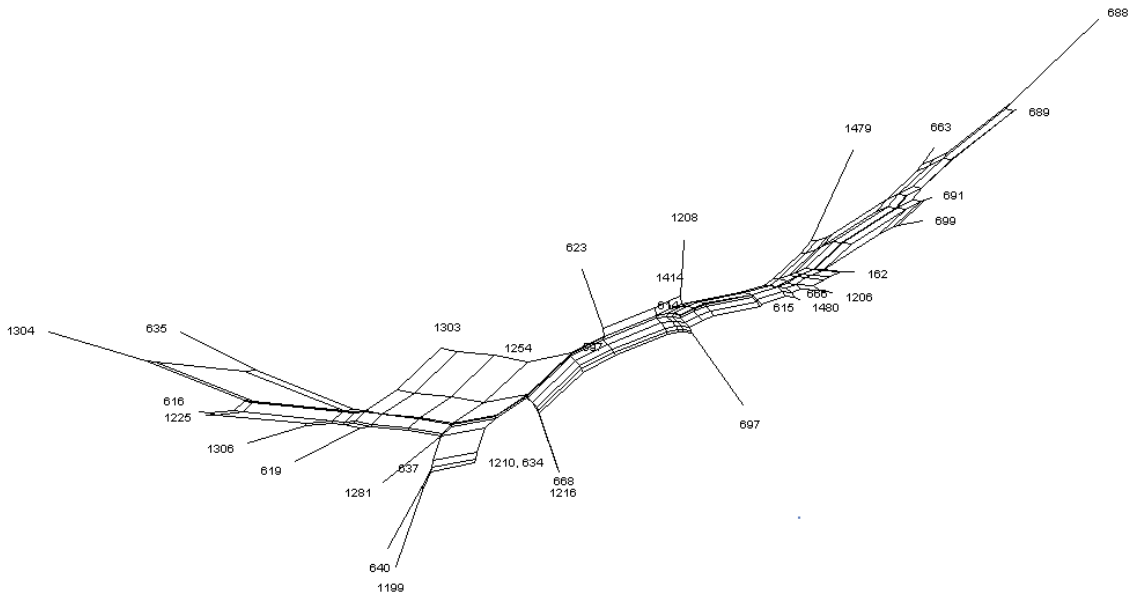
F.3.3. ábra. *Az ötödik dallamosztály elemei intervallum-különbözőségi vektorok távolsága alapján*



F.3.4. ábra. *Az ötödik dallamosztály elemei kronotonikus távolság alapján*



F.3.5. ábra. *Az ötödik dallamosztály elemei vetített kronotonikus távolság alapján*



F.3.6. ábra. *Az ötödik dallamosztály elemei cserélési távolság alapján*